Programowanie Komputerów 4

Sprawozdanie końcowe z projektu zaliczeniowego

Prowadzący - Dr inż. Michał Piela

Temat: Battleship

Maciej Fajlhauer

INF-KTW, grupa 1, sekcja 2

30.06.2024

Analiza tematu:

Battleship to klasyczna gra planszowa, w której dwaj gracze umieszczają swoje statki na planszy i próbują odnaleźć oraz zatopić statki przeciwnika.
Gra kończy się w momencie, kiedy jeden z graczy zatopi wszystkie statki przeciwnika.
W tej wersji gry, drugim graczem jest komputer. Komputer losuje pozycje w którą chce ‘strzelić’, a następnie sprawdza czy nie oddał tam już strzału.

Do napisania programu "Battleship" wykorzystuję język Python wraz z biblioteką Tkinter oraz modułu SQLite3.

Tkinter jest standardową biblioteką interfejsu użytkownika do języka Python. Zapewnia prosty sposób tworzenia aplikacji okienkowych. Tkinter oferuje wiele wbudowanych elementów interfejsu, takich jak przyciski, etykiety i pola tekstowe, które ułatwiają tworzenie interaktywnego interfejsu dla gry. Wybór tej biblioteki był uargumentowany jej prostotą.

SQLite3 to moduł wbudowanej bazy danych w języku Python, co oznacza, że nie wymaga oddzielnego serwera ani konfiguracji. Jest łatwy w użyciu i doskonale nadaje się do prostych aplikacji, takich jak gra "Battleship", gdzie potrzebujemy przechowywać dane logowania użytkowników oraz wyniki graczy.
Dzięki swojej prostocie i elastyczności SQLite3 jest idealnym wyborem do przechowywania i zarządzania małymi zestawami danych w aplikacjach desktopowych.

Specyfikacja zewnętrzna

Ze względu na pomysł zawierający stworzenie działającego systemu logowania do aplikacji, interfejs użytkownika jest bardzo rozbudowanym zagadnieniem.

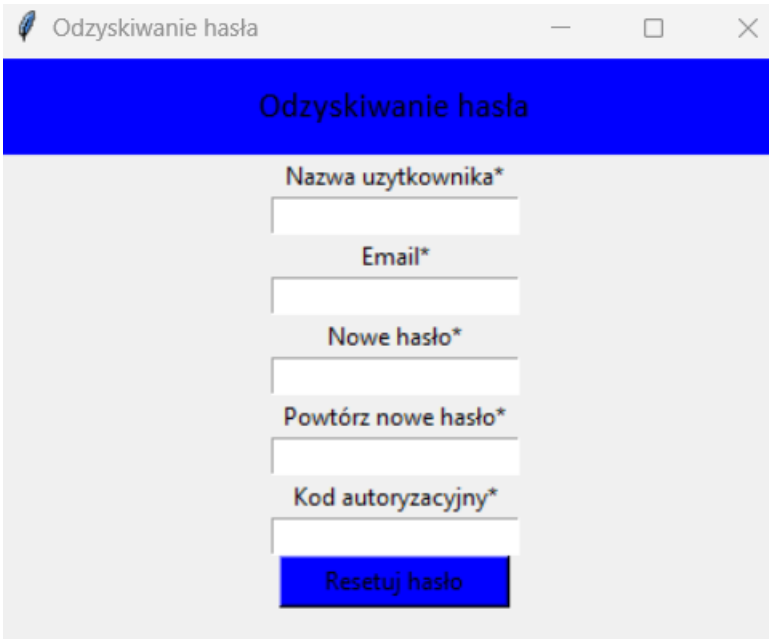
Na samym początku, po włączeniu aplikacji, na ekranie użytkownika pokazuje się pierwsze okno zawierające logo z nazwą gry oraz dwa przyciski (rys.1).

W zależności od sytuacji, użytkownik może się zalogować (rys.2) na już istniejące konto lub je założyć (rys.4), jeśli jest nowym użytkownikiem.

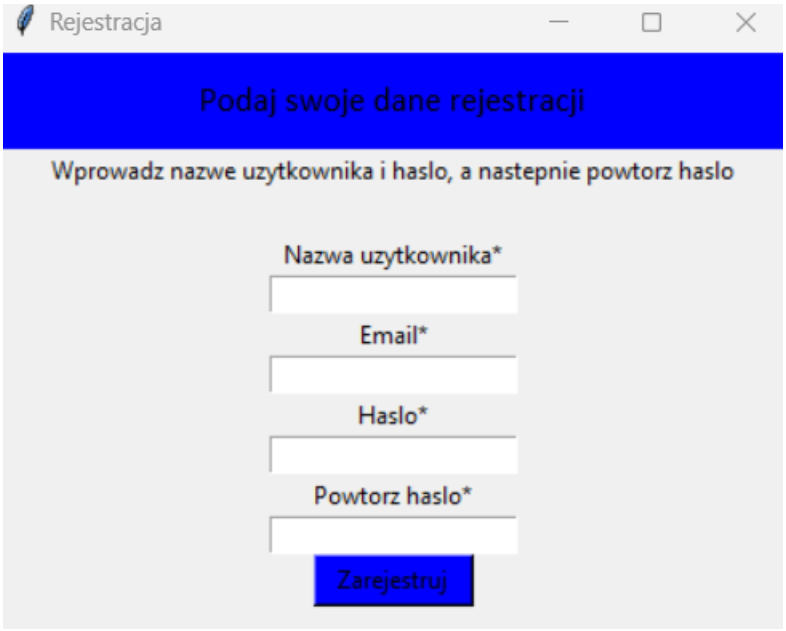
Istnieje również możliwość zmiany hasła, jeśli wystąpi taka potrzeba (rys.3).

Rys.1 Ekran główny

Rys.2 Ekran logowania

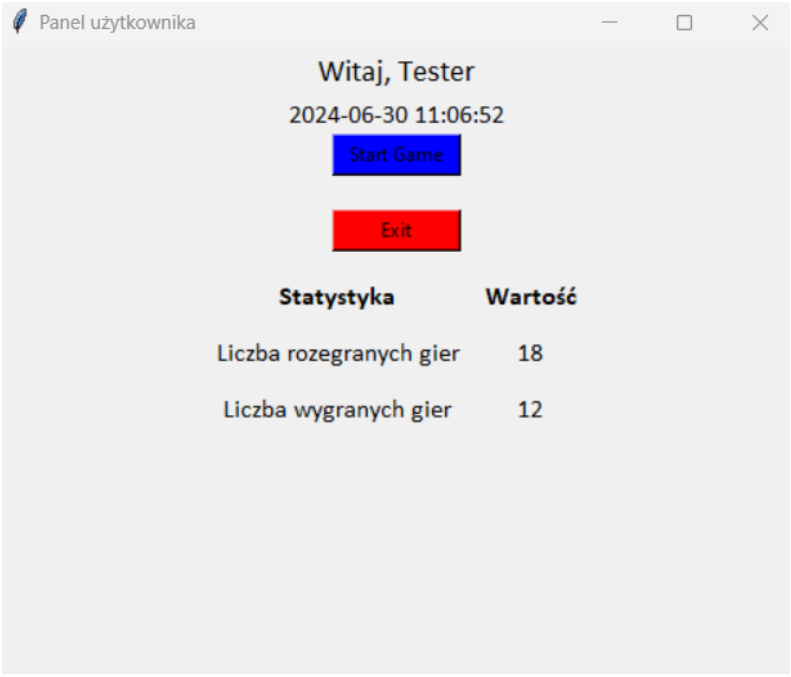


Rys.3 Ekran resetowania hasła



Rys.4 Ekran rejestracji

Po poprawnym zalogowaniu się do aplikacji zostanie wyświetlona główna strona gry. Zawiera ona podstawowe informacje o statystykach bitew, które są pobierane z bazy danych. Oprócz tego użytkownik ma możliwość rozpocząć nową grę lub całkowicie opuścić program (rys.5).

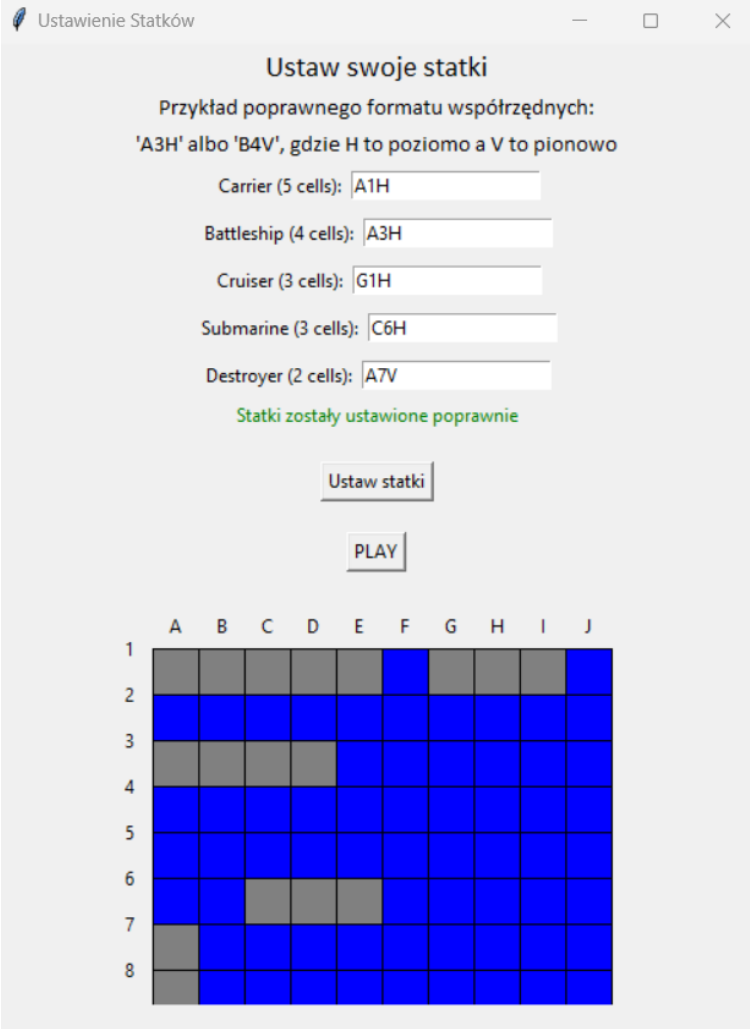


Rys. 5

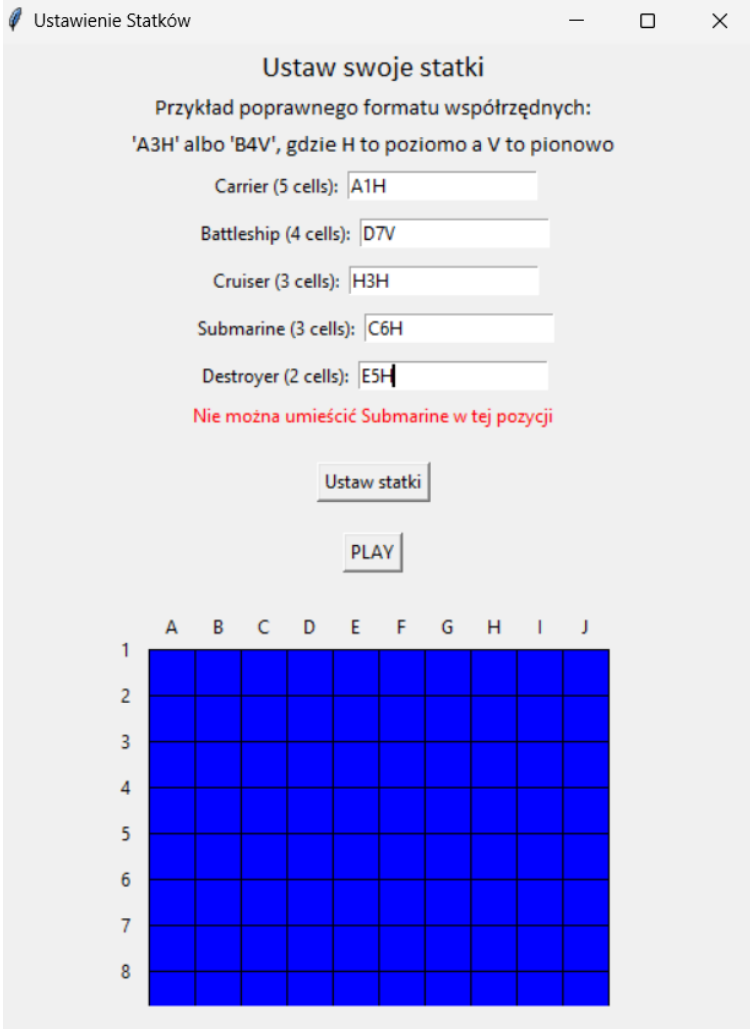
Po wciśnięciu przycisku rozpoczynającego bitwę, przed użytkownikiem ukazuje się nowe okno (rys.6), w którym gracz może ustawić pozycje swoich statków. Orientacja statków jest podawana zaraz po wpisanej pozycji statku. Jest to najprostszy sposób wpisania konkretnych pozycji.

Po wpisaniu prawidłowych pozycji i wciśnięciu przycisku “Ustaw statki”, ich pozycje zostają wyświetlone na planszy poniżej. Wciśnięcie przycisku “Play” przenosi nas do gry z komputerem (rys.8).

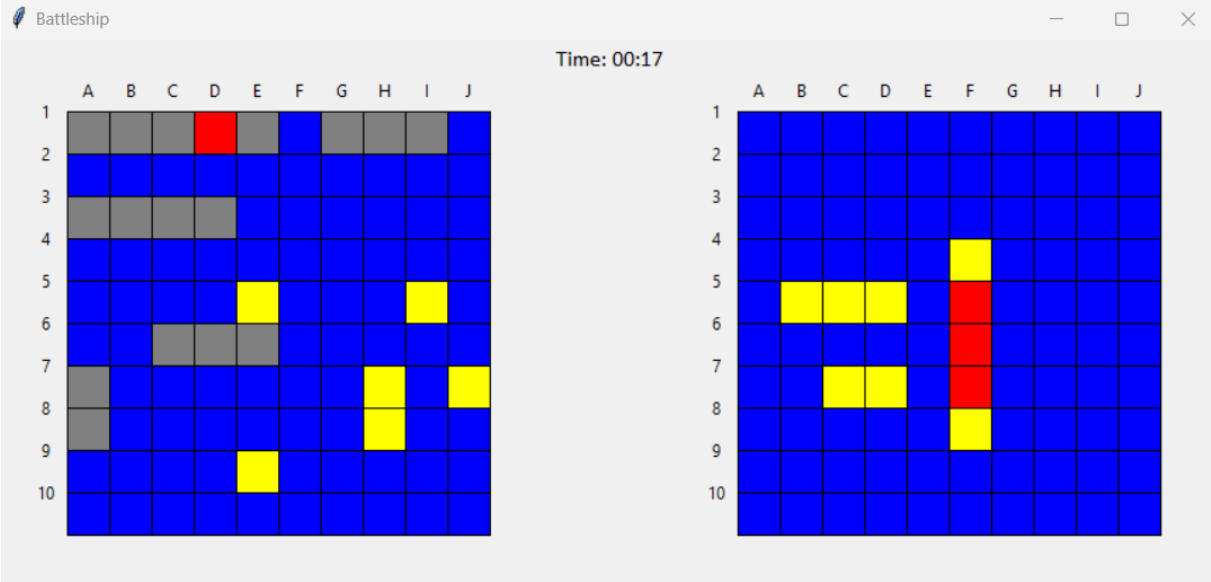
W przypadku podania nieprawidłowych danych, zostaje wyświetlona informacja (rys.7).


Ustawienie Statków
Ustaw swoje statki
Przykład poprawnego formatu współrzędnych:
'A3H' albo 'B4V', gdzie H to poziomo a V to pionowo
Carrier (5 cells): A1H
Battleship (4 cells): A3H
Cruiser (3 cells): G1H
Submarine (3 cells): C6H
Destroyer (2 cells): A7V
Statki zostały ustawione poprawnie
Ustaw statki
PLAY
A B C D E F G H I J
1 2 3 4 5 6 7 8

Rys. 6


Ustawienie Statków
Ustaw swoje statki
Przykład poprawnego formatu współrzędnych:
'A3H' albo 'B4V', gdzie H to poziomo a V to pionowo
Carrier (5 cells): A1H
Battleship (4 cells): D7V
Cruiser (3 cells): H3H
Submarine (3 cells): C6H
Destroyer (2 cells): E5H
Nie można umieścić Submarine w tej pozycji
Ustaw statki
PLAY
A B C D E F G H I J
1 2 3 4 5 6 7 8

Rys. 7



Rys. 8

Plansza po lewej stronie ekranu (rys.8) ukazuje pozycje statków gracza, natomiast po prawej znajduje się plansza przeciwnika. Oddanie strzału odbywa się za pomocą kliknięcia prawym przyciskiem myszy w interesującą nas kratkę.

Statki gracza wyświetlane są na szaro. Jeżeli komputer trafi w nasz statek kratka, w którą został oddany strzał zmienia kolor na czerwony. W przypadku pudła, zmienia kolor na żółty.

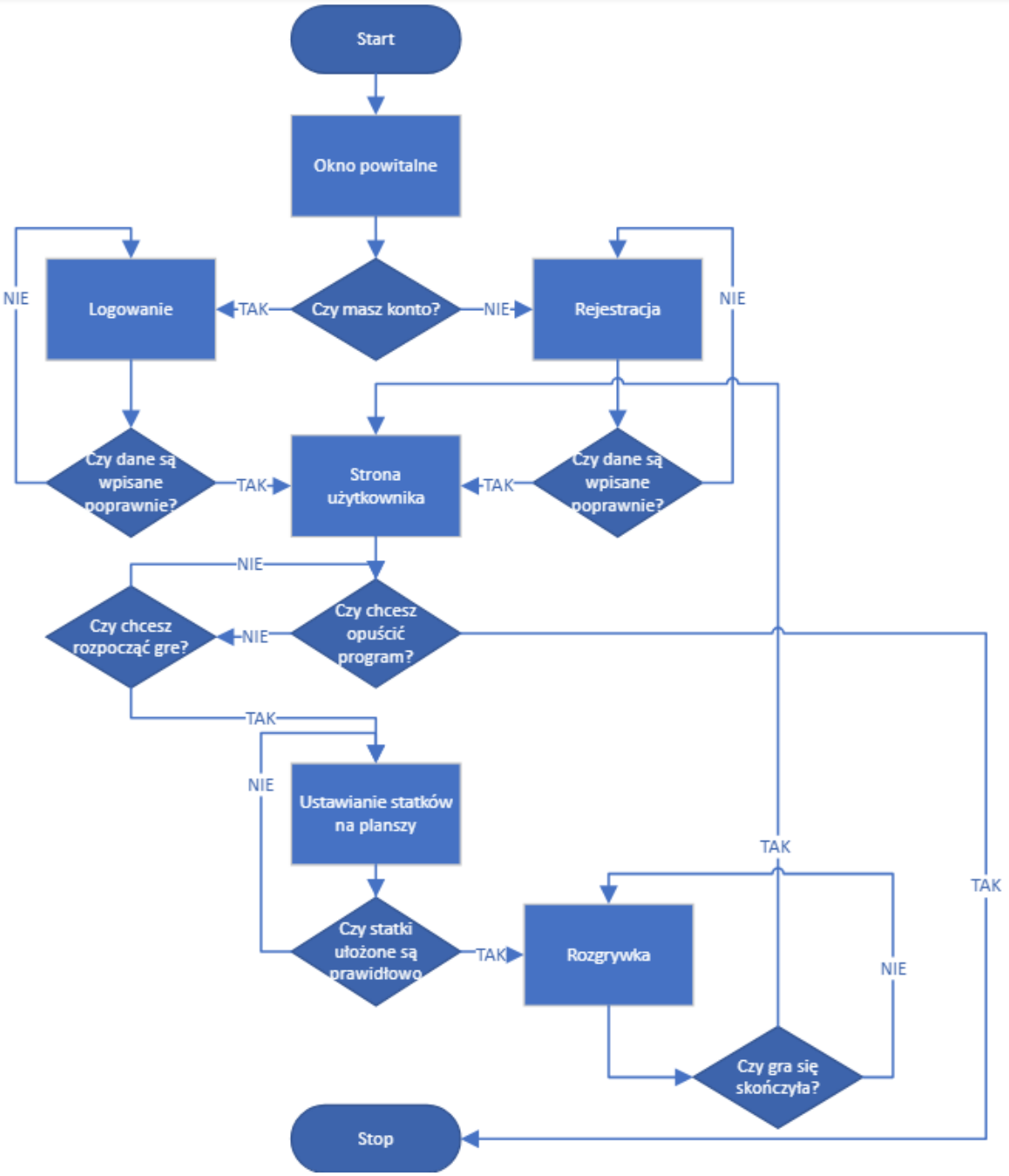
Statki przeciwnika są ukryte. Pudło sygnalizowane jest w ten sam sposób co w przypadku planszy gracza, na żółto. Po trafieniu w część statku przeciwnika, kratka planszy zmienia swój kolor na czerwono. Oprócz tego, użytkownik po każdym trafnym strzale może dodać jeden dodatkowy strzał.

Po zakończeniu gry, czyli kiedy wszystkie statki jednego z przeciwników są zniszczone, program wraca do ekranu głównego oraz zostają zaktualizowane statystki gracza.

W końcowej wersji programu, zostały odrzucone pomysły z przyciskami pozwalającymi na zatrzymanie gry oraz opuszczenie jej przed zakończeniem. Podczas tworzenia programu uznałem, że funkcje te są zbędne.

Specyfikacja wewnętrzna:

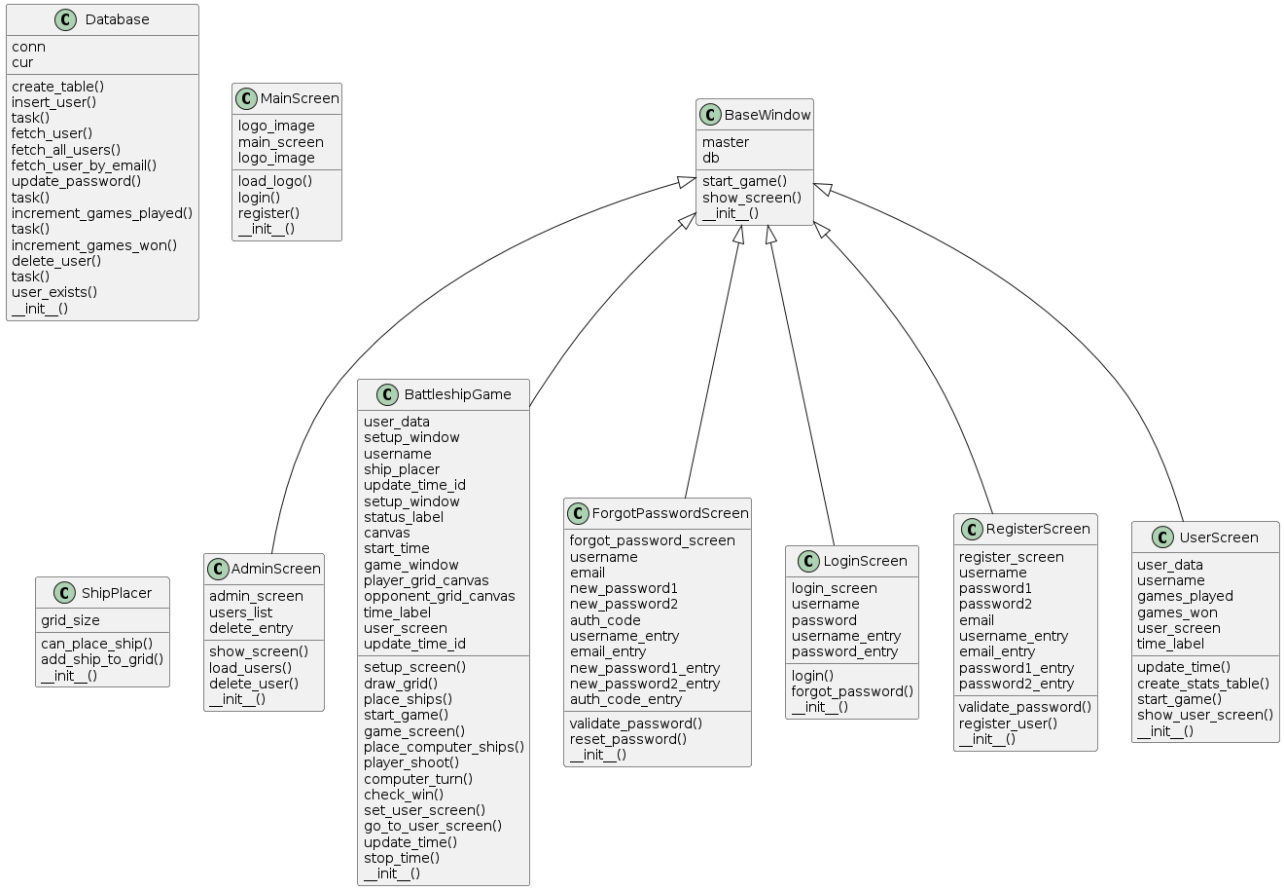
Ogólny schemat działania programu



Rys. 9

Diagram klas

Końcowa wersja programu składa się z 10 klas(rys.10). Sześć z nich dziedziczy po klasie BaseWindow.



Rys.10

Omówienie najważniejszych klas programu:

- **Klasa Database** – realizuje utworzenie bazy, połączenie z nią oraz pozwala na wykonanie takich operacji jak:
    - Dodawanie nowego użytkownika,
    - Aktualizacja hasła,
    - Inkrementacja wyników gry,
    - Usuwanie użytkowników.
- **BaseWindow** – klasa ta powstała w oparciu o **zasadę DRY (Don't Repeat Yourself)**. Zawiera fragmenty połączenia z bazą danych, które powtarzały się aż w 6 klasach.
- **MainScreen** - najważniejsza klasa programu. Dzięki niej użytkownik może zdecydować, czy chce się zalogować czy założyć nowe konto.
- **UserScreen –** klasa UserScreen w głównej mierze odpowiada za wyświetlenie okna użytkownika wraz z jego statystykami oraz pozwala na rozpoczęcie nowej gry.

- **AdminScreen –** pozwala na zarządzanie użytkownikami gry. Udostępnia Adminowi możliwość przejrzenia wyników graczy oraz usunięcia wybranego konta poprzez podanie odpowiedniego numeru id użytkownika.
- **ShipPlacer -** służy do sprawdzania możliwości umieszczenia statku na siatce gry w statki i do faktycznego umieszczania statków na tej siatce.
- **BattleshipGame –** klasa, dzięki której można zagrać w grę "Statki". Zawiera takie metody jak:
    - Utworzenie okna gry wraz ze siatkami i wyświetlanym czasem.
    - Ułożenie statków na planszy i kontrolowanie, aby statki zostały umieszczone poprawnie.
    - Obsługiwanie rundy użytkownika jak i komputera.
    - Sprawdzanie kto wygrał grę.

Techniki obiektowe:

- **Klasy i obiekty –** program opiera się na klasach i obiektach zwiększających poziom abstrakcji
- **Moduły -** program ten podzielony jest na wiele modułów, które pomagają w czytelności programu, ułatwiają rozwój oraz nawigację po elementach projektu
- **Dziedziczenie i polimorfizm –** w końcowej wersji programu powstało wiele klas, które dziedziczą po sobie.

Zagadnienia z zajęć:

- Moduły zostały wykorzystane w celu rozbicia kodu na mniejsze jego fragmenty.
- Wyrażenia regularne (odpowiednik w pythonie: re) zostały użyte w części programu odpowiedzialnego za rejestrację jak i logowanie do programu.
- Ranges zostały wykorzystywane do iteracji wewnątrz pętli
- Wielowątkowość (odpowiednik w pythonie: threading) zaimplementowałem w klasie database.

Biblioteki zewnętrzne:

W tym projekcie nie zostały wykorzystane zewnętrzne biblioteki, które wymagałyby ich wcześniejszego zainstalowania. Niemniej jednak w programie zostały użyte takie biblioteki jak:

- **SQLite3 –** do utworzenia oraz zarządzania bazą danych
- **Tkinter –** do stworzenia interfejsu graficznego
- **Random –** do losowania pozycji statków komputera oraz jego strzałów
- **Time –** użyte w celu wyświetlenia aktualnego czasu systemowego w grze
- **Threading -** umożliwia tworzenie i zarządzanie wątkami

Testowanie i uruchamianie:

W trakcie pisania programu napotkałem problemy takie jak:

- Nieuruchamianie się okna użytkownika lub wyświetlanie nieprawidłowych danych. Problem ten był związany ze złym przekazywaniem argumentu przez klasę Login. Zawartość przekazywanego argumentu, którym była nazwa użytkownika była źle ‘czytana’ przez klasę User. Rozwiązanie okazało się banalnie proste, wystarczyło przekazać cały zestaw informacji (lista krotek) o użytkowniku a już w klasie User rozbić to na konkretne zmienne.
- Kod weryfikacyjny. W przypadku zapomnienia hasła lub chęci jego zmiany na podany adres email przychodzi kod weryfikacyjny, który należy wpisać w odpowiednim polu. Ze względu na brak serwera poczty, kod weryfikacyjny zostaje wyświetlony w wyskakującym okienku.
- Losowość trafień komputera. Ze względu na ograniczony czas nie udało mi się stworzyć “mądrzejszego” bota jako przeciwnika. Pomysł polegał na zmniejszeniu zakresu strzału w przypadku trafienia w statek gracza.
- Nachodzące na siebie statki. Dużym problemem do rozwiązania okazało się rozmieszczenie statków przez komputer. Wynikiem tego problemu jest powstanie klasy ShipPlacer, która m.in. kontroluje poprawność ustawienia statków gracza jak i komputera. Dzięki niej statki na siebie nie nachodzą, nie przecinają się ani nie tworzą jednego długiego statku. Od każdego okrętu jest zachowana min. Jedna kratka odstępu.

Uwagi i wnioski

Battleship był moim pierwszym większym projektem w języku Python. Dzięki możliwości napisania tego programu właśnie w tym języku, zobaczyłem dużo różnic nie tylko w składni, lecz także w sposobie implementacji niektórych elementów jak i w sposobie przekazywania argumentów. Wiedza zdobyta na zajęciach przydała się w trakcie tworzenia gry oraz zachęciła do szukania alternatyw zagadnień laboratoryjnych w innym języku programowania. Projekt ten zamierzam rozwijać dalej w trakcie wakacji. Moim celem będzie zaimplementowanie elementów, których nie zdążyłem zrobiłem podczas trwania tego semestru, tj. poprawienie “strzałów” komputer, czy może nawet wysyłanie kodu weryfikacyjnego przez email.

Ogólny schemat blokowy działania programu (rys.9) powstał przy użyciu aplikacji Microsoft Visio.

Diagram klas programu (rys.10) został wygenerowany przez rozszerzenie PyUML Generator a następnie dzięki stronie internetowej PlantUML.