Politechnika Świętokrzyska

Wydział Elektrotechniki, Automatyki i Informatyki

Karol Rogoziński

Maciej Świercz

Statki

Projekt zespołowy

na studiach stacjonarnych

o kierunku Informatyka

Opiekun projektu:

dr inż. Grzegorz Słoń

Kielce, 2018

SPIS TREŚCI

1. CHARAKTERYSTYKA ZADANIA

Celem projektu było wykonanie aplikacji do gry w statki. Należało przygotować wersję do gry dla dwóch graczy oraz wersję do gry z komputerem. Wersja z komputerem musiała zawierać algorytm, który ułatwiłby mu wygrywanie, czyli m.in. odpowiednie wybieranie pól po trafieniu, ale niezatopieniu statku.

Do wykonania projektu wykorzystano język Java oraz biblioteki graficzne AWT[1] i Swing[2]. Środowiskiem, w którym napisano kod, jest Eclipse[3], ale aplikacja działa pomyślnie na wszystkich platformach IDE Javy, tj. także na Netbeans oraz Intellij.

2. PODSTAWA TEORETYCZNA

Statki to gra polegająca na trafianiu we wszystkie statki przeciwnika. Każdy statek jest złożony z określonej liczby pól - od jednego do czterech. Zatopienie następuje, gdy trafi się wszystkie jego pola. Rozgrywkę wygrywa gracz, który jako pierwszy zatopi wszystkie statki przeciwnika.

W przypadku tej aplikacji pole gry wynosi 10x10, czyli pól jest łącznie 100. Na planszy należy położyć jeden statek zawierający cztery pola, dwa statki wielkości trzech pól, trzy statyki o dwóch polach i cztery statki jednopolowe.

Choć ogólnie gra operuje na tablicy 10x10, to stan pól jest zapisywany w tablicy dwuwymiarowej 12x12. Służy to optymalizacji gry komputera, by ten mógł wykrywać krawędzie i nie próbował trafiać pól, których na planszy nie ma(a jest to możliwe, gdy trafi, ale nie zatopi statku przy krawędzi).

Na początku gracz musi ustawić statki na swojej planszy(po lewej w interfejsie). Aplikacja posiada zabezpieczenia, które sprawiają, że nie można statków ustawić bezpośrednio koło siebie. W czacie gry(w dolnej części interfejsu) wyświetlają się odpowiednie komunikaty, które pomagają graczowi w ustawieniu statków - wyświetla się między innymi informacja ile statków danego typu musi ustawić.

Kiedy statki zostaną już ustawione, zaczyna się rozgrywka. Domyślnie pierwszy ruch wykonuje gracz po stronie serwera. Każde pole po trafieniu może zabarwić się na trzy kolory:

- niebieski - okręt nietrafiony, pole puste
- zielony - okręt trafiony, ale niezatopiony
- czarny - okręt trafiony i zatopiony

Domyślnie każde pole jest szarego koloru.
Po zatopieniu wszystkich statków rozgrywka się kończy, a w czacie gry pojawia się komunikat o tym informujący.

2.1. Gra z komputerem – zasada działania

- **Ustawienie okrętów przez komputer**

Za grę z komputerem w aplikacji odpowiada funkcja „ komputer() ”.

Funkcja ta opiera się na rekurencji ponieważ odwołuję się sama do siebie, dlatego bardzo często wywoływane w niej są różnego rodzaju funkcje sprawdzające zatopienie okrętów czy koniec gry. Na początku należy wyjaśnić oznaczenia przyjęte w programie:

- **„0”** – puste pole, które nie zostało jeszcze użyte,
- **„1”** – pole na którym został postawiony okręt,
- **„2”** – pole na którym był postawiony okręt ale zostało ono trafione,
- **„3”** – pole, które było wcześniej puste jednak zostało trafione tzw.”pudło”,

Podczas gdy gracz ustawia swoje okręty, komputer losuję ustawienie swoich okrętów spośród kilku konfiguracji (rys. 1).

```
if(pierwsze_ustewienie_komputera==0)
{
int los = losuj.nextInt(3); // losujemy z zakresu 0 1 2 3
if(los==0)
{
  int[][] Pole0 = {
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0 },
          {0 , 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  };

  PolePrzeciwnikaInt = Pole0;
  ilosc_przeciwnik_int=20;
  pierwsze_ustewienie_komputera++;
}

if(los==1)
{
  int[][] Pole1 = {
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0 },
          {0 , 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0 },
          {0 , 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  };

  PolePrzeciwnikaInt = Pole1;
  ilosc_przeciwnik_int=20;
  pierwsze_ustewienie_komputera++;
}

if(los==2)
{
  int[][] Pole2 = {
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0 },
          {0 , 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0 },
          {0 , 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0 },
          {0 , 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  };

  PolePrzeciwnikaInt = Pole2;
  ilosc_przeciwnik_int=20;
  pierwsze_ustewienie_komputera++;
}

if(los==3)
{
  int[][] Pole3 = {
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0 },
          {0 , 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0 },
          {0 , 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
          {0 , 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  };

  PolePrzeciwnikaInt = Pole3;
  ilosc_przeciwnik_int=20;
  pierwsze_ustewienie_komputera++;
 }
}

sprawdz_koniec_gry(PoleGraczaInt);
```

Rys. 1. Konfiguracja okrętów komputera

Na potrzeby aplikacji stworzono cztery różne konfiguracje ustawień okrętów komputera ponieważ jak wcześniej zaznaczono funkcja „komputer()” jest funkcją rekurencyjną to w pierwszym kroku należy sprawdzić czy komputer ustawił już swoje okręty.

Jeśli „pierwsze_ustwienie_komputera == 0”, to komputer przypisuje wylosowaną wartość od 0 do 3 do zmiennej typu całkowitego „los”.

Kiedy określona konfiguracja została wylosowana następuje przepisanie tych wartości na faktyczne pole gry komputera oznaczone w kodzie jako

„PolePrzeciwnikaInt” ponieważ właśnie ta tablica zostanie przekazana dla gracza. Kolejnym krokiem jest poinformowanie gracza, że komputer ustawił już wszystkie swoje okręty. Odbywa się to po przez przypisanie do zmiennej całkowitej „ilość_przeciwnik_int” wartości 20.

Ostatnią czynnością jaką należało określić w kodzie przy ustawianiu okrętów przez komputer jest inkrementacja zmiennej „pierwsze_ustawienie_komputera”, aby funkcja przy kolejnym wywołaniu samej siebie nie ustawiała po raz kolejny okrętów.

- **Algorytm gry komputera**

Algorytm gry komputera opiera się na funkcji rekurencyjnej „komputer()”, która wielokrotnie losuje współrzędne „x” i „y” (rys. 2). Współrzędne te przyjmują wartości zgodne z zakresem tablicy od 0 do 9.

```
x = losuj.nextInt(10);
y = losuj.nextInt(10);
```

Rys. 2. Losowanie współrzędnych

Kiedy współrzędne po raz pierwszy zostaną wylosowane do wyboru są dwie opcje komputer może trafić pole na którym został ustawiony okręt gracza i przypisać do niego wartość równą „2” oraz pokoloruje je na zielono (rys. 3).

```
if(PoleGraczaInt[y+1][x+1]==1) // Oznacza ze komputer trafil i ruch zostaje po jego stronie
{
    PoleGraczaInt[y+1][x+1]=2; // oznacz okret jako trafiony
    PoleGracza[x][y].setBackground(Color.GREEN);
    sprawdz_zatopienie_jednomasztowca(x,y);
    sprawdz_zatopienie_trzymasztowca_poziomo(x, y);
    sprawdz_zatopienie_trzymasztowca_pionowo(x, y);
    sprawdz_zatopienie_czteromasztowca_pionowo(x, y);
    sprawdz_zatopienie_czteromasztowca_poziomo(x, y);
    sprawdz_koniec_gry(PoleGraczaInt);
    algorytmy_pomocnicze(x,y);
```

Rys. 3. Komputer trafia okręt

W przeciwnym wypadku, gdy komputer nie trafi okrętu pole o wylosowanych wcześniej współrzędnych przyjmuje wartość równą „3” i kolorowane jest na niebiesko, a ruch zostaje oddany przeciwnikowi (rys. 4).

```
if (PoleGraczaInt[y+1][x+1]==0) // Oznacza ze komputer nie trafil i musi oddac ruch
{
    PoleGracza[x][y].setBackground(Color.BLUE);
    PoleGraczaInt[y+1][x+1]=3;
    ruch_int=0;
}
```

Rys. 4. Komputer nie trafia okrętu

Jeśli jednak komputer trafi okręt co przedstawia kod na (rys. 3), to musi on sprawdzić gdzie pole o danych współrzędnych się znajduje i w zależności od tego podjąć kolejne działania (rys. 5).

```
if(x>0 && x<9 && y>0 && y<9) // Algorytm dla okretów które nie zostały trafione przy krawedziach planszy
{
    ... Odpowiedni kod
}

if(x==0 && y==0) // sprawdzenie lewy gorny rog
{
    ... Odpowiedni kod
}

if(x==0 && y==9) // sprawdzenie prawy gorny rog
{
    ... Odpowiedni kod
}

if(x==9 && y==0) // sprawdzenie lewy dolny rog
{
    ... Odpowiedni kod
}
if(x==9 && y==9) // sprawdzenie prawy dolny rog
{
    ... Odpowiedni kod
}
if(y==0 && x>0 && x<9) // sprawdzenie lewa krawedz
{
    ... Odpowiedni kod
}
if(y==9 && x>0 && x<9) // sprawdzenie prawa krawedz
{
    ... Odpowiedni kod
}
if(y>0 && y<9 && x==0) // sprawdzenie górna krawedz
{
    ... Odpowiedni kod
}
if(y>0 && y<9 && x==9) // sprawdzenie dolna krawedz
{
    ... Odpowiedni kod
}
```

Rys. 5. Sprawdzenie położenia wylosowanego pola

Kiedy komputer ustali położenie wylosowanego pola zostają wykonane kolejne działania zostanie to wyjaśnione na przykładzie pola trafionego w lewym górnym rogu (rys. 6).

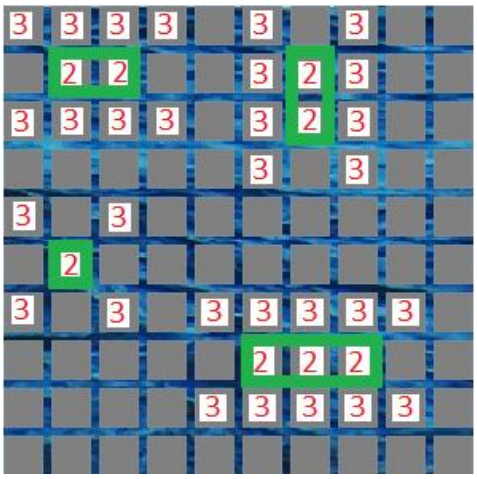
```
if(x==0 && y==0) // sprawdzenie lewy gorny rog
{

    sprawdz_zatopienie_dwumasztowca_w_prawo(x, y);
    sprawdz_zatopienie_dwumasztowca_w_dol(x, y);
    wybor = losuj.nextInt(1); // losuj sposrod 0 lub 1
    if(wybor==0) // sprawdzenie  dol
        {
            if(PoleGraczaInt[y+1][x+2]==1 || PoleGraczaInt[y+1][x+2]==2)
            {
                PoleGraczaInt[y+1][x+2]=2;
                PoleGracza[x+1][y].setBackground(Color.GREEN);
                sprawdz_zatopienie_dwumasztowca_w_dol(x,y);
                sprawdz_zatopienie_trzymasztowca_poziomo(x, y);
                sprawdz_zatopienie_trzymasztowca_pionowo(x, y);
                sprawdz_zatopienie_czteromasztowca_pionowo(x, y);
                sprawdz_zatopienie_czteromasztowca_poziomo(x, y);
                algorytmy_pomocnicze(x,y);
                ruch_int=1;
                sprawdz_koniec_gry(PoleGraczaInt);
                komputer();
            }
            else if(PoleGraczaInt[y+1][x+2]==0)
            {
                PoleGracza[x+1][y].setBackground(Color.BLUE);
                PoleGraczaInt[y+1][x+2]=3;
                ruch_int=0;
            }
            else if(PoleGraczaInt[y+1][x+2]==2 || PoleGraczaInt[y+1][x+2]==3) // Jeśli wylosowane pole było już "używane" został oddany strzał
            {
                komputer();
            }
        }
if(wybor==1) // sprawdzenie  w prawa strone
    {
        if(PoleGraczaInt[y+2][x+1]==1 || PoleGraczaInt[y+2][x+1]==2)
        {
            PoleGraczaInt[y+2][x+1]=2;
            PoleGracza[x][y+1].setBackground(Color.GREEN);
            sprawdz_zatopienie_dwumasztowca_w_prawo(x,y);
            sprawdz_zatopienie_trzymasztowca_poziomo(x, y);
            sprawdz_zatopienie_trzymasztowca_pionowo(x, y);
            sprawdz_zatopienie_czteromasztowca_pionowo(x, y);
            sprawdz_zatopienie_czteromasztowca_poziomo(x, y);
            algorytmy_pomocnicze(x,y);
            ruch_int=1;
            sprawdz_koniec_gry(PoleGraczaInt);
            komputer();
        }
        else if(PoleGraczaInt[y+2][x+1]==0)
        {
            PoleGracza[x][y+1].setBackground(Color.BLUE);
            PoleGraczaInt[y+2][x+1]=3;
            ruch_int=0;
        }
        else if(PoleGraczaInt[y+2][x+1]==2 || PoleGraczaInt[y+2][x+1]==3) // Jeśli wylosowane pole było już "używane" został oddany strzał
        {
            komputer();
        }
    }
```

Rys. 6. Komputer trafił pole w lewym górnym rogu

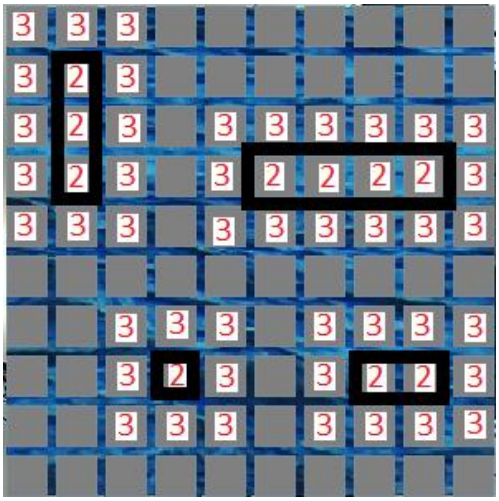
Komputer trafia okręt, więc ruch zostaje po jego stronie w kolejnym kroku musi on wybrać czy chce spróbować trafić okręt w prawą stronę czy w dół ponieważ tylko takie opcje ma w tym przypadku (pole w lewym górnym rogu). Odbywa się to po prze zmienną typu całkowitego „wybor", zgodnie z tym przypadkiem może ona przyjąć dwie wartości „0" gdy komputer sprawdza pole w dół lub „1" gdy komputer sprawdza pole w prawą stronę. Jeśli sprawdzane pole zostaje trafione to przypisuje mu się wartość „2" i koloruje na zielono, następnie zostaje wywoływana rekurencyjnie funkcja „komputer()" ponieważ ruch jest nadal po stronie komputera. Jeżeli zaś wyznaczone przez wybór pole jest puste to przypisuje mu się wartość „3" i ruch zostaje oddany przeciwnikowi.

Kod przedstawiony na (rys. 6) zawiera wiele funkcji pomocniczych, które ulepszają grę komputera, są to algorytmy pomocnicze wykorzystywane przy zatapianiu okrętu oraz przy samej grze komputera tak aby nie losował on współrzędnych na których nie można ustawić okrętów na przykład w przypadku gdy komputer trafi dwa pola w okręcie trzy lub czteromasztowym. Szczegółowo koncepcję algorytmów pomocniczych przedstawia (rys. 7).



Rys. 7. Przykłady algorytmów pomocniczych

Natomiast działanie zatapiania okrętów przedstawia (rys. 8).



Rys. 8. Przykłady zatapiania okrętów

Funkcja „komputer()” wywołuję funkcje pomocnicze bardzo często ponieważ na bieżąco sprawdzane są przedstawione powyżej przypadki, za każdym razem wywoływana jest też funkcja sprawdzająca koniec gry jeśli któryś gracz jako pierwszy zatopi wszystkie okręty w oknie chatu wyświetlany jest odpowiedni komunikat a ruchy obu graczy są blokowane.

3. OPIS DZIAŁANIA APLIKACJI

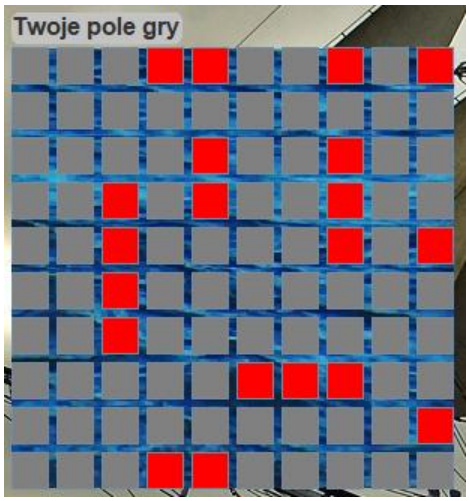
Po odpaleniu programu ukazuje się menu gry (rys. 1). Do wybory są trzy opcje. Po wybraniu gry z komputerem pokazuje się jedno okno i tylko na nim można pracować. Jeśli chce się zagrać z drugim graczem, to po wybraniu opcji gry z graczem należy odpalić także klasę „KlientGra”, aby włączyć drugie okno do gry.

Rys. 1. Okno menu gry

W czacie gry dostaniemy podpowiedzi, jakie statki należy układać (Rys. 2). W trybie przeciwko drugiemu graczowi możemy tutaj także wysyłać wiadomości.

Rys. 2. Czat gry

Po lewej stronie w „Twoje pole gry” (Rys. 3) należy ułożyć wszystkie statki.

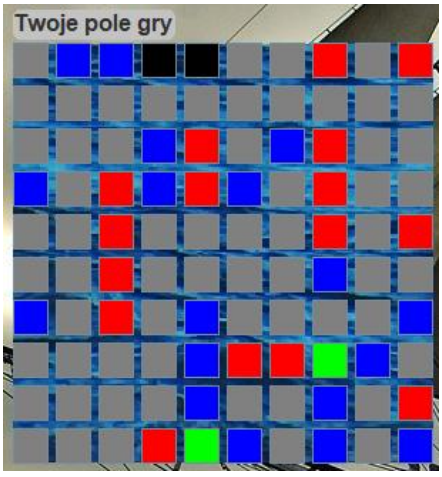


Rys. 3. Lewa strona interfejsu

Po ustawieniu statków można przystąpić do strzelania w planszę przeciwnika - do tej akcji wybieramy pola na planszy po prawej stronie (Rys. 4). Niezatopione, ale trafione statki oznaczone będą kolorem zielonym. Zatopione na czarno.

Rys. 4. Prawa strona interfejsu

W tym czasie także na planszy po lewej stronie zaczną pojawiać się czarne i zielone punkty - to strzały przeciwnika, które są zaznaczane także po lewej stronie (Rys. 5).



Rys. 5. Lewa strona interfejsu po strzałach przeciwnika

4. PODSUMOWANIE I WNIOSKI

Największym problemem, podczas pisania aplikacji, okazał się algorytm gry komputera. To tutaj natrafiono na największe problemy, by upłynnić grę AI. Celem było stworzenie takiego algorytmu, który pozwoliłby na wygrywanie komputera na takich samych szansach jak użytkownikowi - przy odrobienie szczęścia. Należało przy tym rozwiązać problemy związane z wybieraniem odpowiednich pól przez komputer. Trzeba było wziąć pod uwagę kilka przypadków, np. takie, w których komputer trafi, ale nie zatopi statku znajdującego się przy krawędzi mapy. Należało tak go naprowadzić, by wiedział, w którą stronę oddać kolejne strzały. Oczywiście gdy nie trafi żadnego statku, to pole jest wybierane losowo.

Aplikacja - pomimo posiadania obu możliwych trybów gry - posiada ścieżki rozwoju, które można wskazać przyszłym twórcom podobnych aplikacji.

Przede wszystkim można usprawnić odpalanie programu do aplikacji okienkowej włączanej z pulpitu lub dowolnego dysku, a nie ze środowiska Javy. Ułatwiłoby to na pewno wygodę użytkownikowi.

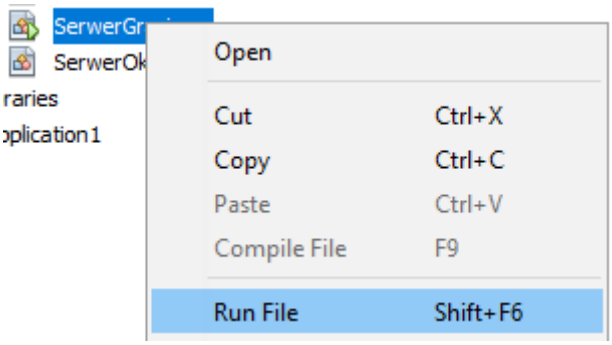
Ponadto w kolejnych wersjach można zaimplementować rankingi oraz punktację, która zliczałaby punkty za zatopione statki. Dzięki takiemu rozwiązaniu użytkownicy mogliby rozgrywać turnieje między sobą i prowadzić statystyki.

Jeszcze innym pomysłem jest wprowadzenie plansz różnej wielkości. Wielkość planszy mógłby ustalać użytkownik przy odpalaniu programu i od tego zależałyby także wielkości statków ustawianych na polach. Jest to niestety dość trudne, bo należałoby taką opcję wziąć pod uwagę przy rozpoczęciu pisania kodu, aby można było elastycznie pisać algorytmy. W tym przypadku nie było to niestety możliwe. Taka opcja na pewno jeszcze bardziej urozmaiciłaby grę i pozwoliła próbowanie różnych wersji gry.

To tylko trzy główne pomysły na usprawnienie gry. Oprócz tego można oczywiście dodać kilka mniejszych szczegółów, jak np. kolorowanie tekstu na czacie czy nawet wybieraniu na nim nicku. Aplikacja pomimo braków tych dodatków jest czytelna i grywalna, a algorytmy w niej zawarte na pracę AI czy rozgrywkę spełniają swoją rolę.

5. INSTRUKCJA OBSŁUGI APLIKACJI

1. Uruchom klasę „SerwerGry”(w przypadku NetBeansa, można użyć skrótu Shift+F6).

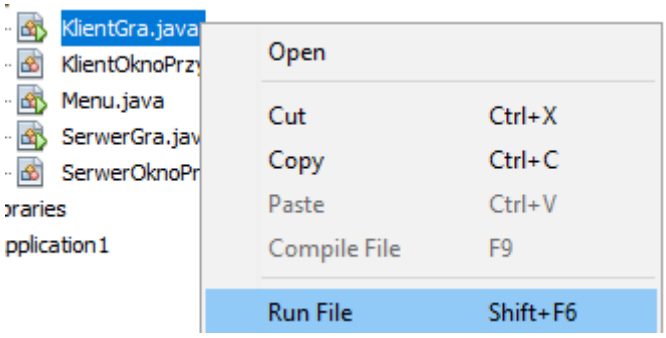


Rys. 6. Uruchomienie klasy

2. Wybierz tryb gry z menu.

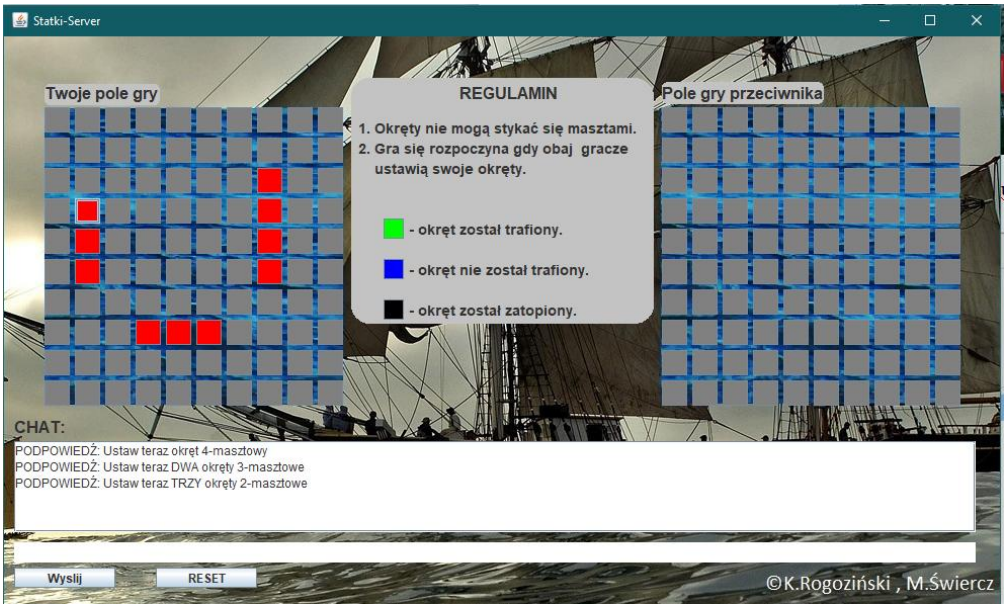
Rys. 7. Panel menu

2.1. Jeżeli wybrałeś grę z drugim graczem, uruchom plik „KlientGra”.



Rys. 8. Uruchomienie klasy do gry z drugą osobą

3. Ustaw statki po lewej stronie. Czytaj komunikaty w czacie, aby dowiedzieć się, jakie statki obecnie należy ułożyć. Po ułożeniu wszystkich statków, zacznij grę próbując trafić statki przeciwnika na polach po prawej stronie.



Rys. 9. Interfejs gry

6. LITERATURA

[1] - Oracle, Oficjalna Dokumentacja biblioteki AWT
https://docs.oracle.com/javase/7/docs/api/java/awt/package-summary.html (dostęp: 01.06.2018).

[2] - Oracle, Oficjalna dokumentacja biblioteki Swing
https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html (dostęp: 01.06.2019)

[3] - Eclipse Foundation, oficjalna strona programu IDE
https://www.eclipse.org/downloads/ (dostęp: 01.06.2018)