Dokumentacja - Gra "Statki kosmiczne"

Ernest Bieś, Konrad Czechowski, Dawid Kwaśny

8 czerwca 2019

Spis treści

1 Podstawowe informacje

"Statki kosmiczne" to gra logiczna będąca połączeniem dwóch popularnych gier "Statki" oraz "Saper".

2 Zasady gry

Po rozpoczęciu gry na planszy o wymiarach 9x9 utworzonej z kwadratów są losowo rozmieszczone statki kosmiczne o różnych wielkościach:
Dwa *Transportowce* o wymiarze 4
Trzy *Samoloty kosmiczne* o wymiarze 3
Trzy *Wahadłowce* o wymiarze 2
Dwa *Szturmowce* o wymiarze 1

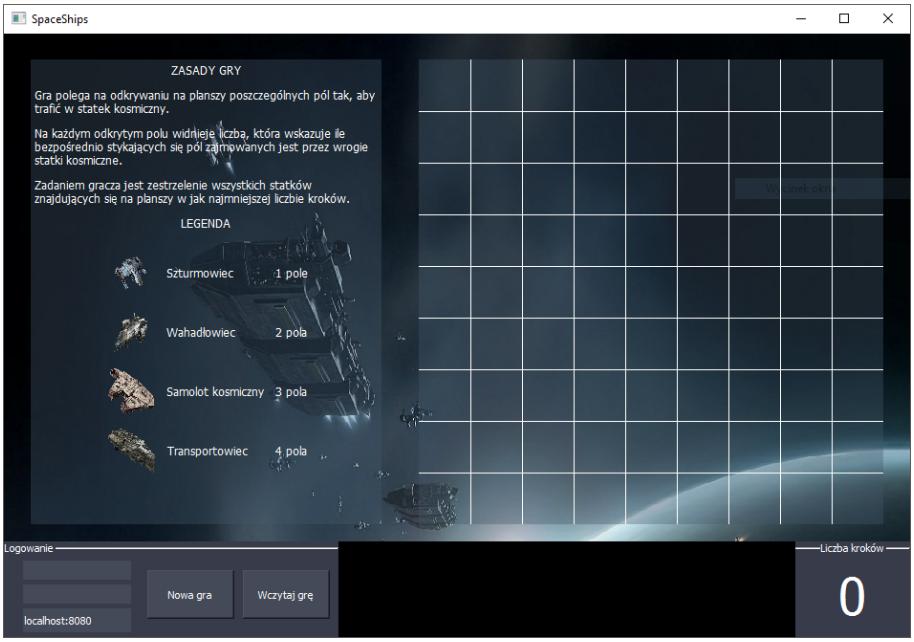
Statki nie są widoczne. Zadaniem gracza jest odnalezienie wszystkich statków w jak najmniejszej liczbie kroków. W przypadku odkrycia pola, które nie zawiera żadnego statku widoczna jest liczba, która wskazuje na ilu polach sąsiadujących z odkrytym polem znajdują się statki. Wobec powyższego gracz musi podejmować swoją decyzję dotyczącą pola, które w następnej kolejności odkryje na podstawie logicznego myślenia, gdyż tym samym zwiększa swoje szanse na odnalezienie statków w pobliżu.

3 Opis klienta

3.1 Środowisko programistyczne

Klient gry "Statki kosmiczne" został utworzony z wykorzystaniem darmowego środowiska programistycznego QtCreator 4.9.1.

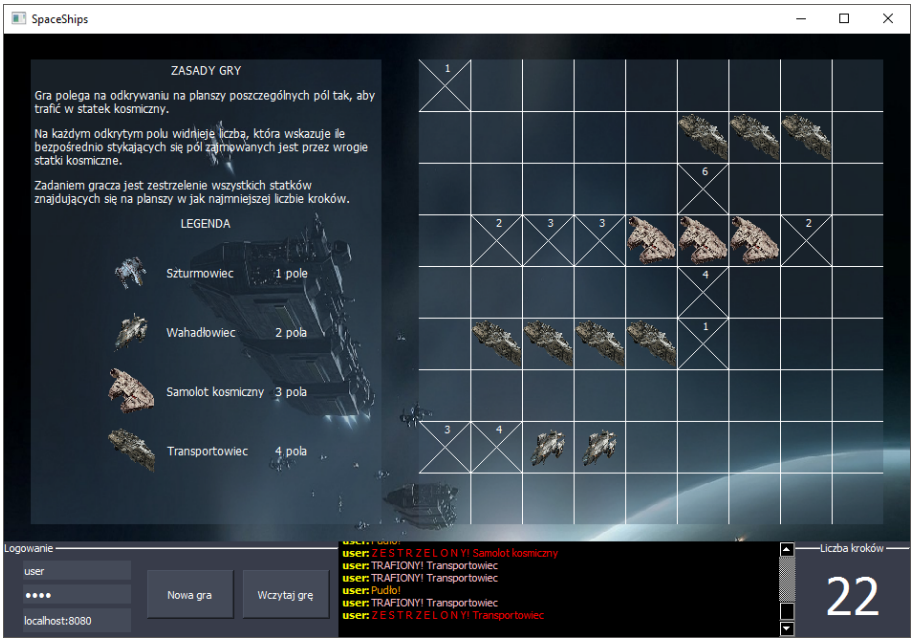
3.2 Interfejs graficzny klienta



Rysunek 1: Okno główne programu

W głównym oknie programu po lewej stronie widoczna jest plansza o rozmiarze 9x9 kwadratów na której wyświetlane są ikony zestrzelonych statków oraz pola, które zostały odkryte przez gracza, a nie zawierają żadnych statków. Po uruchomieniu gry po lewej stronie wyświetlone zostają podstawowe informacje dotyczące zasad gry oraz statków jakie musimy zestrzelić. W górnym lewym rogu znajduje się przycisk za pomocą którego możemy ukryć zasady gry. W lewym dolnym rogu okna znajdują się pola służące do wprowadzenia nazwy użytkownika i hasła oraz przyciski za pomocą których możemy utworzyć nową grę na serwerze lub wczytać już istniejącą.Na dole w srodku okna znajduje się konsola na której wyświetlane są informacje do-

tyczące akcji podejmowanych przez gracza. Z prawej strony w dolnej części okna znajduje się licznik z aktualnie wykonanymi krokami przez gracza.

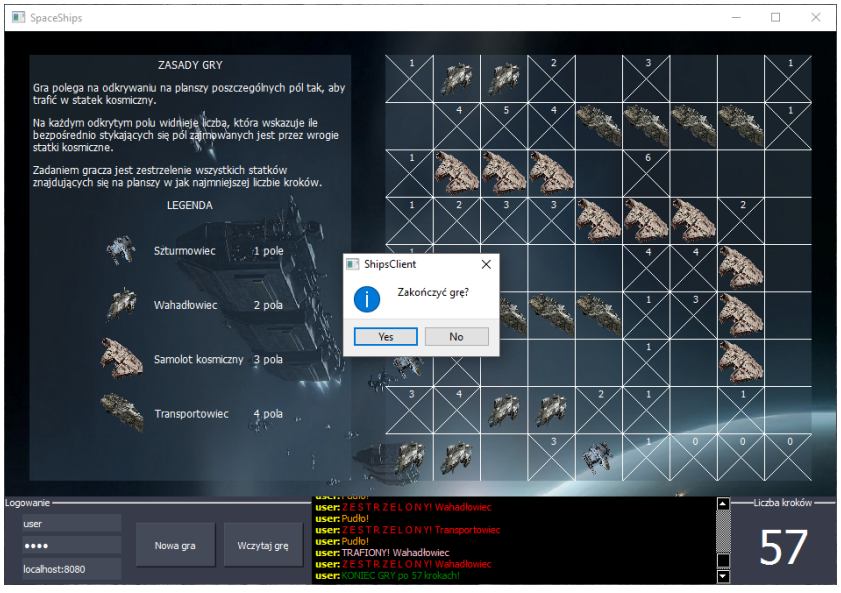


Rysunek 2: Okno w trakcie gry

W trakcie gry użytkownik po wskazaniu myszką wybranego pola i naciśnięciu lewego przycisku myszki odsłania poszczególne pola na planszy. W przypadku trafienia statku jego ikona zostaje wyświetlona w odpowiednim miejscu na planszy, natomiast w przypadku "pudła" we wskazanym polu pojawia się znak X oraz liczba wskazująca na ilu polach sąsiadujących ze wskazanym znajdują się statki.Równocześnie po wskazaniu pola odtwarzany jest odpowiedni dźwięk.

3.3 Instrukcja obsługi

Po uruchomieniu klienta użytkownik wprowadza nazwę użytkownika oraz hasło, a następnie naciska przycisk "Nowa gra". Zostaje utworzona nowa gra dla użytkownika, następnie korzystając z myszki naciska lewy przycisk na wybranym polu na planszy. Jeżeli pod odkrytym polem znajduje się statek, jego ikona zostanie wyświetlona w odpowiednim kwadracie oraz program zasygnalizuje trafienie odpowiednim dźwiękiem. Jeżeli natomiast wskazane pole nie zawiera żadnego statku to zostanie na nim wyświetlony znak X oraz liczba wskazująca na ilu polach sąsiadujących znajdują się statki. Jednocześnie za każdym wskazaniem pola zostaje zwiększony licznik znajdujący się w prawym, dolnym roku ekranu. Jednocześnie na konsoli będą pojawiały się komunikaty dotyczące trafienia danego statku, jego nazwy lub też, że oddany strzał to niestety "pudło".
Po zestrzeleniu wszystkich statków kosmicznych wyświetlany jest komunikat z zapytaniem czy użytkownik chce zakończyć działanie programu. W przypadku udzielenia negatywnej odpowiedzi gra jest zapisywana na serwerze, natomiast klient zeruje wszelkie dane i ustawienia celem umożliwienia użytkownikowi rozpoczęcia nowej gry.



Rysunek 3: Okno po zakończeniu gry

4 Analiza problemu

5 Implementacja

Rozdział 5

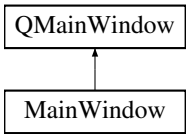
Dokumentacja klas

5.1 Dokumentacja klasy MainWindow

Klasa MainWindow definiuje metody tworzące i obsługujące sesje gry na serwerze, odpowiada za całościowy wygląd klienta gry.

`#include <mainwindow.h>`

Diagram dziedziczenia dla MainWindow



Metody publiczne

- **MainWindow** (QWidget ∗parent=nullptr)
- Status newGame (QString user, QString server)

  Metoda wywoływana podczas tworzenia nowej gry. Wysyła żądanie do serwera w celu utworzenia nowej sesji gry. Odtwarza dźwięk.
- Status shotGame (QString user, QString shot, QString server)

  Metoda wywoływana podczas kliknięcia na jedno z pól na planszy. Wysyła żądanie do serwera w celu sprawdzenia i zapisania rezultatu oddanego strzału.
- Status getGame (QString user, QString server)

  Metoda wywoływana podczas wczytywania gry. Wysyła żądanie do serwera w celu otrzymania danych o istniejącej sesji gry.
- void showStatus (Status ∗status)

  Wypisuje informacje do konsoli. Odtwarza odpowiednie dźwięki. Wyświetla zapytania do użytkownika.

Metody chronione

- void mousePressEvent (QMouseEvent ∗event)

  Rysuje znak X lub ikonę statku na kliknięty polu.
- void paintEvent (QPaintEvent ∗)

  Przeciążenie rysowania zapewnia skalowanie tła.
- void resizeEvent (QResizeEvent ∗event)

  Ustala położenie wszystkich elementów w skalowalnym oknie.

5.1.1 Opis szczegółowy

Klasa MainWindow definiuje metody tworzące i obsługujące sesje gry na serwerze, odpowiada za całościowy wygląd klienta gry.

5.1.2 Dokumentacja funkcji składowych

5.1.2.1 getGame()

```
Status MainWindow::getGame (
            QString user,
            QString server )
```

Metoda wywoływana podczas wczytywania gry. Wysyła żądanie do serwera w celu otrzymania danych o istniejącej sesji gry.

Parametry

user	nazwa użytkownika
server	adres serwera gry

Zwraca

5.1.2.2 mousePressEvent()

```
void MainWindow::mousePressEvent (
            QMouseEvent * event )  [protected]
```

Rysuje znak X lub ikonę statku na klikniętym polu.

Parametry

event	

5.1.2.3 newGame()

```
Status MainWindow::newGame (
            QString user,
            QString server )
```

Metoda wywoływana podczas tworzenia nowej gry. Wysyła żądanie do serwera w celu utworzenia nowej sesji gry. Odtwarza dźwięk.

Parametry

user	nazwa użytkownika
server	adres serwera gry

Zwraca

Status

5.1.2.4 shotGame()

```
Status MainWindow::shotGame (
            QString user,
            QString shot,
            QString server )
```

Metoda wywoływana podczas kliknięcia na jedno z pól na planszy. Wysyła żądanie do serwera w celu sprawdzenia i zapisania rezultatu oddanego strzału.

Parametry

user	nazwa użytkownika
shot	kod współrzędnych pola
server	adres serwera gry

Zwraca

Obiekt Status

5.1.2.5 showStatus()

```
void MainWindow::showStatus (
            Status * status )
```

Wypisuje informacje do konsoli. Odtwarza odpowiednie dźwięki. Wyświetla zapytania do użytkownika.

Parametry

status	

Dokumentacja dla tej klasy została wygenerowana z plików:

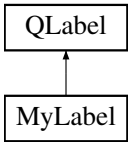
- mainwindow.h
- mainwindow.cpp

5.2 Dokumentacja klasy MyLabel

Klasa MyLabel opisuje obiekt QLabel wyświetlający planszę gry.

```
#include <mylabel.h>
```

Diagram dziedziczenia dla MyLabel



Metody publiczne

- MyLabel (QWidget ∗parent)

  Konstruktor dziedziczący po QLabel, zapewniający takie metody jak paintEvent.

Metody chronione

- void paintEvent (QPaintEvent ∗)

  rysuje pole gry, siatkę, tło

5.2.1 Opis szczegółowy

Klasa MyLabel opisuje obiekt QLabel wyświetlający planszę gry.

5.2.2 Dokumentacja konstruktora i destruktora

5.2.2.1 MyLabel()

```
MyLabel::MyLabel (
            QWidget * parent )  [explicit]
```

Konstruktor dziedziczący po QLabel, zapewniający takie metody jak paintEvent.

Parametry

parent	

Dokumentacja dla tej klasy została wygenerowana z plików:

- mylabel.h
- mylabel.cpp

5.3 Dokumentacja klasy Status

Klasa Status zapewnia metody służące do komunikacji klienta z serwerem. Przechowuje ilość oddanych strzałów, nazwę trafionego statku, otrzymaną planszę z serwera i inne.

```
#include <status.h>
```

Metody publiczne

- Status (QString code, QString shipName, int type, int steps, QString board)

  Konstruktor tworzący nowy obiekt Status zgodnie z podanymi parametrami.
- int getSteps ()

  Zwraca liczbę kroków zapisanych w obiekcie Status.
- QString getCode ()

  getCode
- QString getShipName ()

  Zwraca nazwę trafionego statku.
- int getType ()

  Zwraca typ trafionego statku.
- QString getBoard ()

  Zwraca planszę gry w formie tekstowej.
- QString toString ()

  Parsuje obiekt Status do ciągu tekstowego w formacie JSON do wysłania do serwera.
- void setSteps (int steps)

  Ustawia ilość kroków.
- void setCode (QString code)

  Ustawia kod trafionego pola.
- void setShipName (QString shipName)

  Ustawia nazwę trafionego statku.
- void setType (int type)

  Ustawia typ trafionego statku.
- void setBoard (QString board)

  Ustawia stan planszy gry w formie zapisu tekstowego.

5.3.1 Opis szczegółowy

Klasa Status zapewnia metody służące do komunikacji klienta z serwerem. Przechowuje ilość oddanych strzałów, nazwę trafionego statku, otrzymaną planszę z serwera i inne.

5.3.2 Dokumentacja konstruktora i destruktora

5.3.2.1 Status()

```
Status::Status (
            QString code,
            QString shipName,
            int type,
            int steps,
            QString board )
```

Konstruktor tworzący nowy obiekt Status zgodnie z podanymi parametrami.

Parametry

kod	
nazwa	statku
typ	
ilość	kroków
plansza	

5.3.3 Dokumentacja funkcji składowych

5.3.3.1 getBoard()

```
QString Status::getBoard ( )
```

Zwraca planszę gry w formie tekstowej.

Zwraca

plansza

5.3.3.2 getCode()

```
QString Status::getCode ( )
```

getCode

Zwraca

kod odpowiedzi np. SHOTDOWN, ENDGAME

5.3.3.3 getShipName()

```
QString Status::getShipName ( )
```

Zwraca nazwę trafionego statku.

Zwraca

nazwa trafionego statku

5.3.3.4 getSteps()

```
int Status::getSteps ( )
```

Zwraca liczbę kroków zapisanych w obiekcie Status.

Zwraca

liczba kroków

5.3.3.5 getType()

```
int Status::getType ( )
```

Zwraca typ trafionego statku.

Zwraca

typ trafionego statku

5.3.3.6 setBoard()

```
void Status::setBoard (
            QString board )
```

Ustawia stan planszy gry w formie zapisu tekstowego.

Parametry

plansza	

5.3.3.7 setCode()

```
void Status::setCode (
            QString code )
```

Ustawia kod trafionego pola.

Parametry

kod,np	"ENDGAME"

5.3.3.8 setShipName()

```
void Status::setShipName (
            QString shipName )
```

Ustawia nazwę trafionego statku.

Parametry

nazwa	statku

5.3.3.9 setSteps()

```
void Status::setSteps (
            int steps )
```

Ustawia ilość kroków.

Parametry

ilość	kroków

5.3.3.10 setType()

```
void Status::setType (
            int type )
```

Ustawia typ trafionego statku.

Parametry

typ	trafionego statku

5.3.3.11 toString()

```
QString Status::toString ( )
```

Parsuje obiekt Status do ciągu tekstowego w formacie JSON do wysłania do serwera.

Zwraca

obiekt Status w formie JSON

Dokumentacja dla tej klasy została wygenerowana z plików:

- status.h
- status.cpp

6 Testowanie

7 Proponowane dalsze możliwości rozbudowy

8 Instrukcja kompilacji

Oprócz standardowej kompilacji z poziomu środowiska QtCreator możemy skompilować program ręcznie. Poleceniem *qmake* należy wygenerować plik Makefile. Następnie wystarczy uruchomić polecenie *make*, a po chwili w katalogu pojawi się skompilowany plik wykonywalny.