Contents

1 Presentation **1**

2 Contents **1**

3 Code documentation **1**

4 Source code compilation **1**

5 Playing BattleShip2D **2**
5.1 Overview 2
5.2 Positioning the player's ships 2
5.3 Playing against the computer 3

1 Presentation

BattleShip2D is a Java-FX game to play to the classic BattleShip in graphical version: one human user plays versus the computer and each protagonist targets the cell of the other one, every turn. The winner is the first protagonist which has destroyed the entire fleet of opponent's vessels.

2 Contents

This file is part of the **BattleShip2D** project, made of:

- *Documentation*: provides
  - this file, both in LaTeX and PDF formats;
  - *doxyfile*, the `Doxygen` configuration file used to produce code documentation in HTML. In order to get drawings of classes relationships, the **Graphviz** package is required.
  - *html* and *latex* directories, resulting from running `Doxygen`.
  - *figures*, grouping images used in this document.
- the *META-INF* directory used to create *BattleShip2D.jar*
- the *COPYING* file provided to comply to GPL3 license
- the jar archive named *BattleShip2D.jar*. Besides the code classes, this archive contains the source code of the `model` package.
- *README.TEXT*, a short notice about the project's contents.

3 Code documentation

Both *html* and *latex* directories should contain the complete project code documentation (either in HTML or LaTeX formats), resulting from running `Doxygen` on the *doxyfile* file. However, if this directory either does not exist or is empty, you can re-create the documentation files by running:

```
$ doxygen
```

4 Source code compilation

This manual should be provided in the directory *BattleShip2D*. This directory should also contain Java files located in the subdirectory `battleship2d`. WARNING: the subdirectory `battleship2D/ui` only contains classes but no `JavaFX` code.

Before compiling, make sure that `Java 8` has been properly installed on your computer.

1. One way to know your version of `java`[1]:

   ```
   $ java -version
   ```

2. Some systems may have several different versions of `Java`. Type the following commands to list them:

   ```
   $ update-alternatives --config java
   ```

3. If `Java 8` is actually in the list but is not selected as actually the used version, the system offers you to select the valid version (you may need to be logged as *root*).

4. Once you have the valid versions of both `java` and `javac`, go into the directory *BattleShip2D* and run the compilation as follows:

   ```
   $ javac -Xlint:all battleship2d/model/*.java
   ```

5. You can also (re)create the jar archive *BattleShip2D.jar* as follows:

   ```
   $ jar cmvf META-INF/MANIFEST.MF BattleShip2D.jar META-INF battleship2d/*
   ```

5 Playing BattleShip2D

The objective for the player is to place its ships on its game board, then play against the computer to destroy the opponent's ships.

5.1 Overview

Run *BattleShip2D.jar* by typing:

```
$ java -jar BattleShip2D.jar
```

or move to the `battleship2D`'s parent directory and type:

```
$ java battleship2D.ui.BattleShip2D
```

The main windows are displayed (fig. 1).

- On the top-left part, the players's board is displayed as a 10x10 empty game board. Each cell is currently empty, with an ocean background.
- On the top-right part, the computer's game bord is currently hidden, until the player has positioned its fleet of ships.
- The bottom part is made of two panels: one for selecting the ships to position on the board, the other for displaying game information. The first panel is a choice box, providing information about fleet ships:
  - Name (**CARRIER**, **BATTTLESHIP**, **CRUISER**, **DESTROYER**, **SUBMARINE**)
  - Size in cells (**5**, **4**, **3**, **2**, **1**)
  - the `Done` button is used to valid ship insertion. The computer's board is displayed as soon as every player's ship has been positioned (an error message is displayed if the button is pushed while some ships have not been positioned).

5.2 Positioning the player's ships

The **Information** panel at the very bottom of the application prompts dynamic messages.

1. The first message is:

   ```
   [Player] Select your ships and place them on your board (mouse button 1 to validate,
   mouse button 3 to cancel).
   ```

2. Inserting a ship follows a 4-step procedure, described in fig. 2. The set of available cells for every ship depends on the location of the center cell (selected by clicking on it) and the ship already placed on the board. Use the Ship insertion choice box to select the kind of ship you want. It is possible to change the position of a ship already placed on the board. And remember that clicking on the mouse button 3 cancels the last ship insertion.

[1] The procedure is the same for `javac` is to type (under `Debian`-like systems)

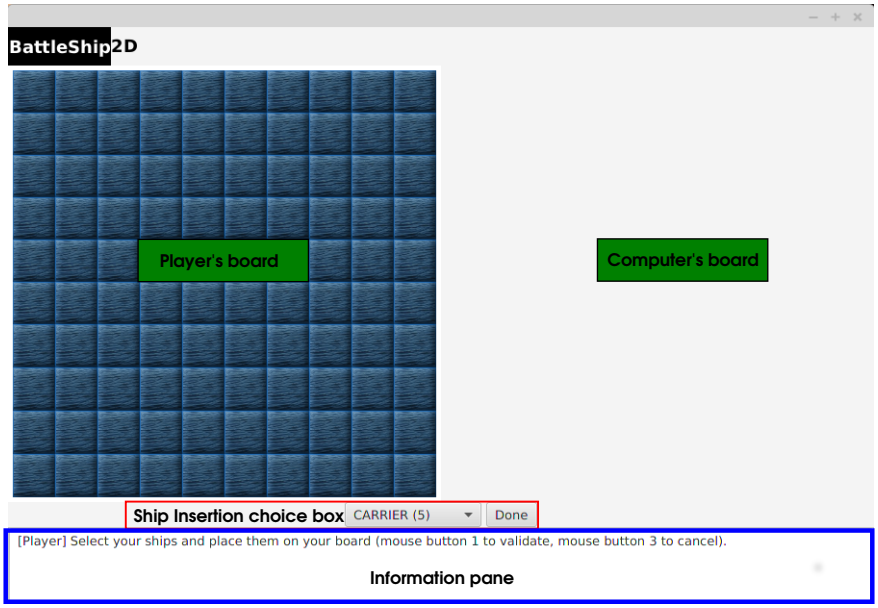

Figure 1: Description of game windows

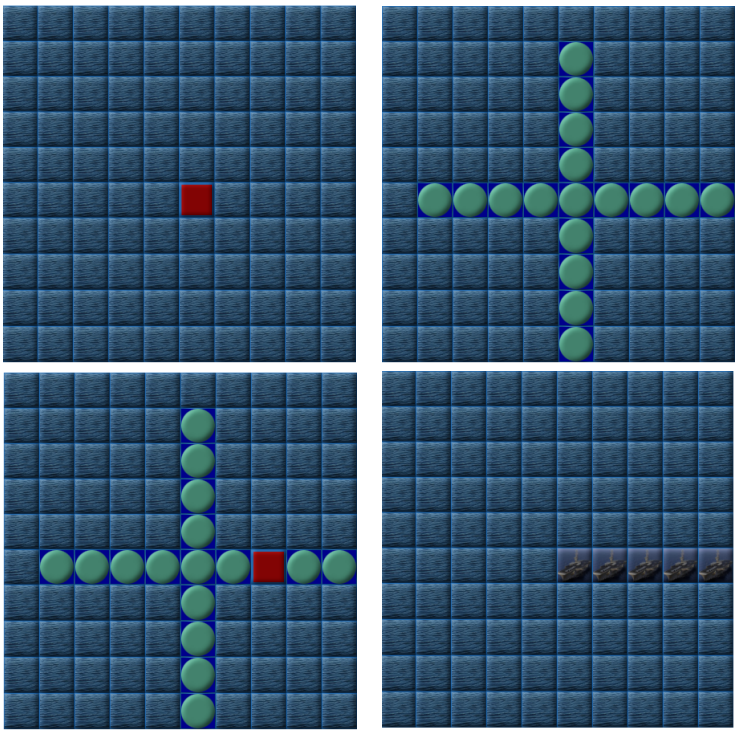

Figure 2: Ship insertion. From top-left to bottom-right: the cell under the mouse is marked in red; after clicking, the cells available to position a ship are displayed along the four cardinal orientations; hovering the mouse on a available cell allows to choose an orientation; clicking once again inserts the current ship.

5.3 Playing against the computer

1. Once all the ships have been inserted, click on the button "`Done`"[2] next to the choice box. The computer's board is diaplayed, with every cell shown in black to hide its ships (fig. 3). A message in the information pane prompts you to select a cell on the computer side.

2. Once a cell has been clicked on the computer side, a missile (displayed as an orange sphere) is launched from one of the player's ships[3], as shown in fig. 4.

[2]This button does nothing while there is at least one ship to insert (and a warning message is displayed in the information pane).
[3]The ship is randomly chosen each time. More precisely, the missile is launched from one of the cell filled with an image ship that has not been hit.

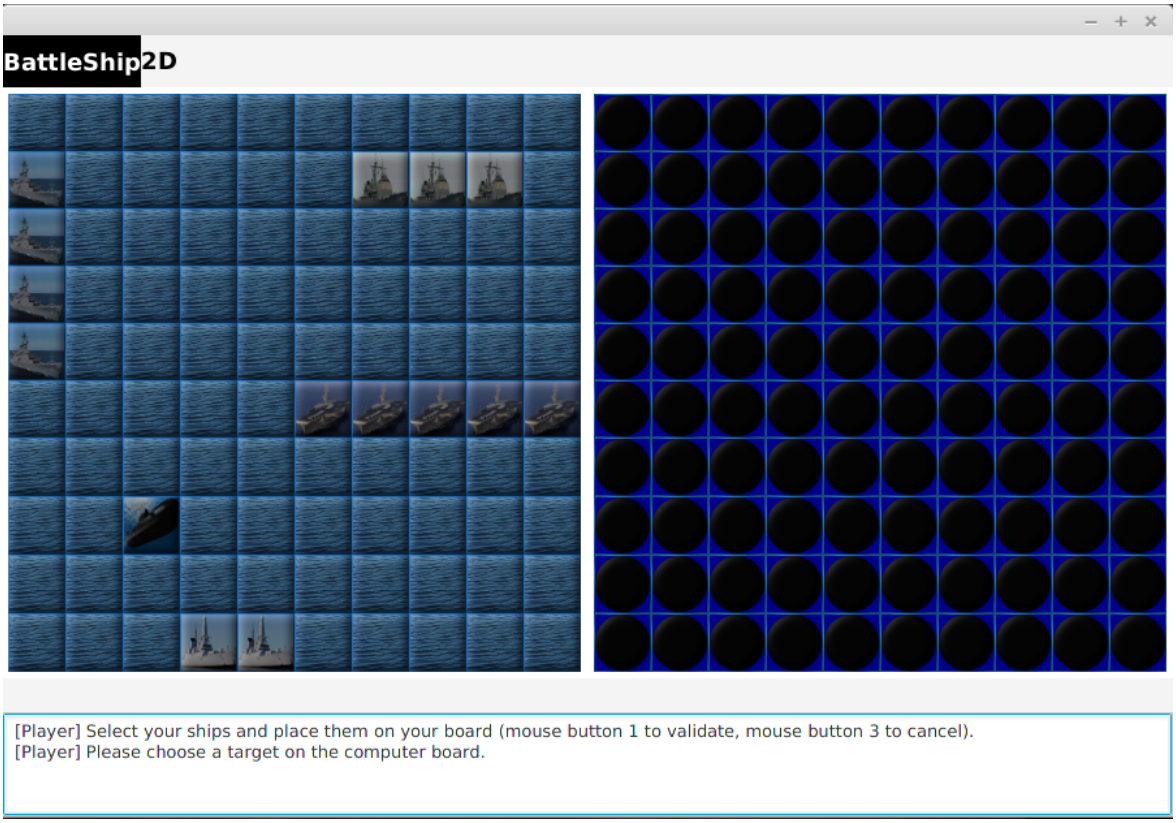

Figure 3: The game is ready to start.

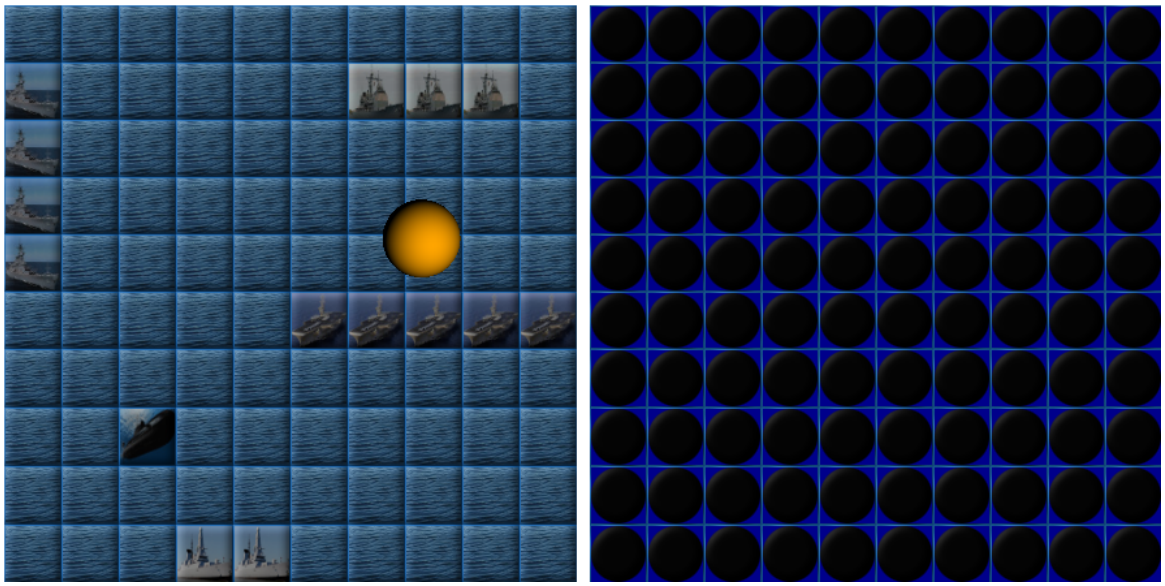

Figure 4: The first missile is launched.

3. The computer fires back with its own missile, and the game turns alternate between player and computer moves. If the missile hit only water, an "ocean" cell is displayed on the computer side. If a ship has been hit, an image of a burning ship is displayed, as shown in fig. 5 on the computer side, after somme unsuccessful attempts from the player. A message is displayed in the information pane each time a ship has been hit.

   The computer plays with a configurable level of expertises, ranging from shooting at randomly cells to select areas next to hit ships on the player side.

4. The game stops as soon as one of the opponents has destroyed all the ships of its adversary. A last animation with the name of the winner is run as soon as the last ship has been destroyed (fig. 6).

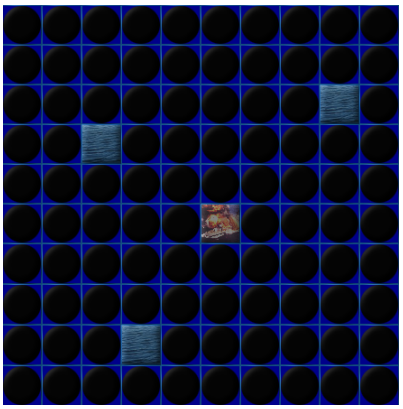

Figure 5: A ship has been hit on the computer side.

Figure 6: Top: the game is almost finished: the player is still looking for the computer's submarine while its only intact ship is a two-cell sized destroyer. Bottom: the player has touched the computer's submarine and wins the game.