Antibattleship

6.005 Project 2 Design

Ned Murphy, Julia Boortz, Justin Venezuela

April 26, 2011

1 Overview

Implementing our Antibattleship (ABS) game will consist of coding three main parts: the *client* code, which represents the state of a game of antibattleship outside of representation and user interaction, the *GUI* code, which represents the user's interaction with the client, and the *network* code, which represents the client's interaction with external players.

2 Client

Any game of ABS will create a new instance of the class ABSGame, which will take in two ABSPlayer instances. ABSGame acts as a message handler between the two players, and controls the flow of the game. Each ABSPlayer, whether representing a human player, an AI player, or a network player, must be able to both send and receive ABSMessages from an ABSGame. ABSGame then determines when an ABSMessage should be sent/received to/from either player. However, ABSPlayers need not know the protocol used to communicate with other players: they need only communicate to ABSGame using our ABSMessage class. ABSGame handles stripping down these ABSMessages into strings that follow the protocol, as well as reconstructing ABSMessages from these strings.

It is worth noting that ABSPlayer's ignorance of the protocol allows for, say, easy implementation of AI players, since we needn't concern ourselves with parsing Strings, etc.. However, this is not too restricting in the space of possible ABSPlayers we can make: one ABSPlayer already written simply acts as a wrapper class for human input into a console, in which the human player is expected to simply type messages following the protocol.

3 GUI

We note that one could play a game of Antibattleship using the client code without a graphical user interface. However, this is surely not a good idea if we have user friendliness in mind. Our GUI interacts with ABSGame and displays the relevant game state information to the user. The GUI code also includes listeners which take user inputs and sends the information to ABSGame.

4 Network

Our network code is responsible for connecting to network players. This code is decoupled from the client. The network code needn't understand our ABSMessage scheme or even the network protocol. It need only be able to connect outside players to ABSGame and handle passing protocol strings between ABSGame and these outside players.

5 Flow Diagrams

We have drawn out flow diagrams that map out the functionality of our ABS game. They are available (as image files) in the SVN in the same folder as this document.

6 Contact

Please direct any questions regarding our design to `nedmurp@mit`, `jboortz@mit`, and `jven@mit`. ■