Design Document

Battleship

Team: Jason Warta and Laurin Fisher

Table of Contents

Introduction

The purpose of this document is to define the design and layout of our project code, including but not limited to the architecture and functions used. This software is designed to allow two humans to play the game of battleship from two separate computers over the web.

System Overview

The goal is to have our project run in a browser with as much functionality of the game on the client side as possible without sacrificing security or functionality. Our project is written in HTML, Javascript, and CSS using the Meteor databasing framework and Jasmine and Velocity as testing frameworks. We designed our code based off a state machine using session (global) variables and an easy-to-use user interface.

System Architecture

The software is organized into three main files, battleship.html, battleship.js, and battleship.css. However structure wise, our code depends on the database software Meteor. There is a client side and server side, any game elements that do not require data passed between users (placing ships, displaying grids, mouse clicks, etc) are to be placed in the client side of meteor. The server side part of the game is to keep game elements such as score, whether a hit misses or hits, and originally logging in and connecting to another user for a game.

Outside of these blocks are Mongo collections that allow us to globally keep track of players, player actions, and board data. We use Session variables, created on render of the game, for a state-machine like layout. When this player is in “placing” mode then we are allowed to click on and place ships on one grid, etc.

Meteor operates on templates, meaning a template is made in the html, and helper functions are then made in the javascript. Meteor links these files together for use automatically and often does little niceties like this for us.

Meteor has helper functions for each template and events for each template. Since our template is ‘game’ we have helper functions that act as regular javascript functions and event functions that activate when we click the mouse etc.

The HTML page is the display, the javascript the power, and CSS the style, with Meteor as the databasing software that holds it all together.

Data Design / Data Structures

The data is stored in Mongo collections, and elements in a collection are called documents. It is helpful to think of the collection as an array of class objects in a language like C++. Mongo collections are synced across the client and server, making them easily accessible from both sides with Meteor. Earlier in this document we mentioned session variables. These variables can be compared to global variables in the sense that they do not go out of scope and can be called by any of our functions for our state machine code design.

Control Hierarchy

As with HTML, Javascript, and CSS the control hierarchy is as follows. The html defines or loads an object for the user to see and interact with. This can be text, drawing boxes, grids, etc. These objects, or elements, are linked to Javascript through the keyword “class” in HTML. The javascript functions with the same name (with the particular event in front= ex:”click. shipSelect”) are then called when the HTML objects are interacted with in some way (clicked on usually). The CSS classes are called if you call them in Javascript with JQuery or from the HTML itself when it is loading images, etc.

Javascript code is only executed if it is placed properly within the meteor blocks. So functions have to be in a meteor template within the helper functions or event functions, within the server or client block, or in the Meteor methods block to be executed. As for testing, Jasmine creates a certain folders for client and server side unit tests as well as integration tests. The test files have to go in there, though the code we are original files do not.

Meteor ultimately, starts running the code through a browser, connecting to the local machine currently as localhost:3000, though we hope to put the game on a server so it is accessible to everyone over the web.

Design Patterns

We organized our code into multiple Javascript and CSS files to follow module-like programming. We also organized our directory to cater to Meteor. Sprite sheets are in the public folder, tests in the test folder, client side code in the client folder, etc. As for the code itself, it is organized with comments and blocks, particularly, javascript functions are stored in the meteor methods block on the server side code. Our other javascript functions are inside our game template either inside the template helper block or the template events block on the client side. Our client code block is currently empty but probably will not stay that way.

On another note, our code is designed similarly to a state machine, in which certain functions are executed when the user has initiated certain states within the game. An example of this would be the state of placing ships. The mouseover function will then have the corresponding ship sprite follow the user’s mouse pointer until they have placed the ship on the grid.

Specifications

As normal, different functions are called at different times, however we have some basic functions we can describe. For starters, one of the first functions called is initCellArray. This javascript function does what it sounds, it initializes a blank grid to the screen for both the user’s board and the user’s guesses on the enemy’s board. It is the first function called on initial render because this is where the beginning of the game starts. Our game template also has several helper functions that return our session variables for easy access (similar to get functions in C+

+ classes). These are called because they are connected to HTML elements. We also have several event functions such as clicking on a cell on the grid or mousing over the grid. These can places ships or show you where your ship will be placed. These are obviously called when an event occurs.

Appendices

This Template was taken and modified from: https://blog.udemy.com/software-design-document-template/

The title image used for educational purposes was taken from: http://worldartsme.com/images/battleship-clipart-1.jpg