System design document for Sänk oss Project (SDD)

1 Introduction

1.1 Design goals

The application has been designed to use as loose couplings as possible. Modules should not be coupled in a way that transpose cyclic dependencies in the domain model. These modules should, in this way, be replaceable, as well as testable independently of each other. For more information regarding usability, see the RAD.

1.2 Definition, acronyms and abbreviations

- RAD - Requirements and Analysis Document
- MVC - Model View Controller-pattern. A way of splitting theapplication into parts, where each part's soul purpose is to handle its area of work. (GUI, game logic and data handling)
- GUI - Graphical User Interface.
- State-pattern - Software design pattern, which allows changing in behaviour of an object.

2. System design

2.1 Overview

The system's design is based on the framework LibGDX, as well as the MVC-pattern, that is a slightly modified MVC-pattern to suit the framework, which has led to a good use of the State-pattern.
For more information about LibGDX, see 2.2.1.

2.1.1 Application Model

As mentioned, the model of the applicaiton has been modified to suit the choosen framework, LibGDX, which has resulted in a hybrid solution based on the well known MVC-pattern and the State-pattern.
Since the applicaiton is set out to use different Screens throughout the game, based on what part of the game the player is in, it's appropriate to be using an abstact representation of a Screen within the Controller class. This abstract representation will, at run-time, be set to a concrete Screen based on user actions. This will furthermore support good use of polymorfism within the application, lower the cohesion and make it easy adding new Screens in the future.

2.1.2 Game Rules

The rules of the game is pre-defined and not at all modified, all to suit users familiar with the well known board game Battle Ships. For rules see References in RAD.

2.1.3 Input Handling

The applications input wont be handled by a single instance of a tradional InputHandler-class, but will vary with the different Screens, i.e, when a new Screen is set the game will change its input source along with the Screen. This is the way the framework is designed and it might seem strange at first, but it has shown to be an effective way of dealing with user input.

With that said, it's worth mentioning that each Screen holds the Event Handling for its Actors and will be told (by them) when to act upon user interaction.
Also, see 2.2.1 for framework details and 2.1.4 for Event Handling specifics.

2.1.4 Event Handling

Network events will be handled inside the Client-class. This class has the ability to set listeners which will be notified when something has happened over the network. This class also has pre-defined methods for sending data to the server.

An interface is supplied to realise a class for handling network events. A class realising this interface is later overridden so that an instance doesn't have to define every method inside the interface.

2.2 External resources

2.2.1 The LibGDX Framework

The application uses the framework LibGDX, which is a light-weight, yet advanced game framework based on OpenGL. Using this framework ensures a true platform independent application.

"libGDX is a cross-platform Java game development framework based on OpenGL (ES) that works on Windows, Linux, Mac OS X, Android, your WebGL enabled browser and iOS." (Sweet, 2014)

The application model, and it's MVC structure is built around the way LibGDX works, as it has Screens with render methods and code to handle the user input.

2.2.2 The KryoNet library

To handle networking, the application uses a library called KryoNet, written by the same author as of LibGDX.

“KryoNet is a Java library that provides a clean and simple API for efficient TCP and UDP client/server network communication using NIO. KryoNet uses the Kryo serialization library to automatically and efficiently transfer object graphs across the network.” (Sweet, 2013)

Both the client and the server share the protocol defined in the core part of the application. This is written to apply to the KryoNet library. It is deeply integrated into both the server and the client.

Sources

Nathan Sweet, 2014-05-13 KryoNet:
https://github.com/EsotericSoftware/kryonet

Nathan Sweet, 2014-05-12 LibGDX:
https://github.com/libgdx/libgdx/blob/master/README.md