System design document for RunningMan (SDD)

Contents

Version: 0.2

Date 2015-05-05

Author Armand Ghaffarpour, Simon Lindkvist, Jesper Nilsson, Johan Tobin

This version overrides all previous versions.

1. Introduction

1.1 Design goals

The design goals are to use libgdx framework and MVC design pattern. The idea is that the model shouldn't know anything about the framework, so that it will be easy to swap. As long as the visual framework provides the tilemap functionality, it can be used. The design should be as modular as possible, so that certain features can be deleted and the application remains runnable.

1.2 Definitions, acronyms and abbreviations

- GUI, graphical user interface
- Java, platform independent programming language
- Libgdx, external library for a game engine
- MVC, Model View Controller is a design that tells you how to partition an application.

2. System design

2.1 Overview

The application will use an MVC model. The model will handle the internal representation of data for a player, enemies and level objects. The visual representations will be handled by the view classes, using the libgdx framework. Libgdx is also used for handling keyboard input in the PlayerController and for reading tile maps of the tmx format in the map handler class.

2.1.1 The model functionality

The game loop is situated in the libgdx render method, which is called upon all the time. This render method updates the GameController, which updates the other controllers, and tells the LevelView and HudView to draw themselves. The PlayerController handles the input from the keyboard and updates the Player model accordingly.

2.1.2 Rules

The Level model handles the time of the level and updates the time. The GameController checks if the time us up, and if it is, quits the level and takes the user to the main menu. The Player model keeps track of if the player is dead or not, and whether the player has finished the level or not, and the GameController checks those states in every update.

2.1.5 Event handling

2.2 Software decomposition

2.2.1 General

Package diagram. For each package an UML class diagram in appendix

2.2.2 Decomposition into subsystems

2.2.3 Layering

2.2.4 Dependency analysis

2.3 Concurrency issues

NA. This is a single thread application. Therefore there are no concurrency issues. The game is however based on events in the controller classes that are determined from the user input.

2.4 Persistent data management

No data is saved between playing sessions. However all objects in the game are inserted into arraylists for management.

2.5 Access control and security

NA

2.6 Boundary conditions

NA - The application will be launched as a normal desktop application.

3. References

1. MVC, see https://developer.apple.com/library/mac/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html
2. Libgdx, see http://libgdx.badlogicgames.com

APPENDIX