System Design Docouments

Gabriel Wallin, Omar Oueidat, Erik Strid, Olof Enström

May 2017

Version

Version: 3.0
Date: 17 May 2017
Author: Revised by Omar Oueidat
This version overrides all previous versions.

Contents

1 Introduction

1.1 Design goals

The main design is to create a program that is loosely coupled. The program should be easily expandable and adding features and complexity should be easy. Also, the design must be testable.

1.2 Definition, acronyms and abbrevations

- Miner - The player
- Hull - The miners health, which decreases when he hits something or crashes
- Fuel - The gas of which the miner uses to be able to move around libGDX - The library used to create the game.
- MVC - Model-View-Controller, a design pattern that organises the program in a structure in a way that avoids high coupling and dependencies
- Drilling - The act of which the player will dig a hole into the ground
- Gear - A collective word for Fuel and Hull

2 System Architecture

2.1 Overview

The design will be structured around a MVC design model, with low coupling and loose connection between classes and packages.

2.2 Listeners

The view will heavily depend on listeners to update the playscreen. The model will send a signal to the view that tells it to update accordingly and send necessary data that the view needs. Having listeners was chosen to decrease the dependencies and lower the coupling between the view and model only letting the view access the information that the model wants it to access.

2.3 Game Screens

To changes between different views of the game, will be handled with screens in which the application can switch between. Having the Start screen and the play screen on different screen. This will keep coupling low since they do not have to communicate with each other.

2.4 libGDX

The program will be constructed around the libGDX library, which is a cross-platform game and visualization development framework.

LibGDX will provide the tools to easily construct the game.

libGDX lets you go as low-level as you want, giving you direct access to file systems, input devices, audio devices and OpenGL via a unified OpenGL ES 2.0 and 3.0 interface

3 Subtype decomposition

3.1 First Software to describe

3.2 Next Software to describe

4 Persisten data management

5 Access control and security

6 References

Denna kör vi också bibliotek med hjälp av IEEEtran