System design document for the Tetrix project (SDD)

Contents

Version 1.0
Date 2011-05-20
Author Linus Karlsson, Jonathan Kara, Andreas Karlberg, Magnus Huttu

This version overrides all previous versions.

1 Introduction

1.1 Design goals

The design must be loosely coupled to make it possible to switch GUI and/or partition the application into a client-server architecture. The design must be testable i.e. it should be possible to isolate parts (modules, classes) for test. For usability see RAD.

1.2 Definitions, acronyms and abbreviations

All definitions and terms regarding the core Tetrix game are defined in the references section.

- GUI, graphical user interface.
- Java, platform independent programming language.
- JRE, the Java Runtime Environment. Additional software needed to run a Java application.

- Host, a computer where the game will run.
- State, the present condition of the system. Each view is considered to be a state.
- MVC, a way to partition an application with a GUI into distinct parts avoiding a mixture of GUI-code, application code and data spread all over.
- Slick, a 2D game library
- FPS, frames per second

2 System design

2.1 Overview

The application uses the framework "Slick". Each state (intro sequence, main menu, gameplay etc.) has three methods in which the state is initiated, rendered and updated - init, render and update. The init method is acting like a constructor and is used to initiate all the values needed. Keypresses and operations are the update method's responsibility, which is called frequently depending on the FPS of the game. The render method is where the updates are drawn on the screen (images, objects).

All the states is a part of Slick's StateBasedGame pattern, where all the states/views are handled by a higher class responsible for initiating states and moving between them.

2.2 Software decomposition

2.2.1 General

The application is decomposed into the following modules, see Figure 1.

- Main is the application entry class
- core, the OO-model
- core.tetrominoes, all the tetrominoes in the OO-model
- util, general utilities
- view, the different states of the program
- sound, sound handling

2.2.2 Layering

The layering is as indicated in Figure 1.

2.2.3 Dependency Analysis

Dependencies are as shown in Domain model in the RAD's appendix. There are no circular dependencies.

2.3 Concurrency Issues

NA. This is a single thread application. Everything will be handled by the Slick framework.

2.4 Persistent data management

All persistent data will be stored in .dat files. The files will be;

- A file for high score (names and scores)
- A file for the player's name

2.5 Access control and security

NA

2.6 Boundary conditions

NA. Application launched and closed as normal desktop application (scripts).

3 References

1. Tetris game: http://en.wikipedia.org/wiki/Tetris
2. MVC, see http://en.wikipedia.org/wiki/Model-view-controller