System design document for Mosquito Simulator

Version: 1.3

Date: 31/05-15

Author: Alexander Sandberg, Johan Ljungberg, Rasmus Davidsson, Anton Solback

This version overrides all previous versions.

1 Introduction

1.1 Design goals

The design must follow the Open-Closed principle so that it's easy to extend the application in the future. The design must be testable i.e. it should be possible to isolate parts (modules, classes) for the test. For usability see RAD.

1.2 Definitions, acronyms and abbreviations

- GUI, graphical user interface.
- Java, platform independent programming language.
- JRE, the Java Runtime Environment. Additional software needed to run an Java application.
- Energy, for the mosquito. The "healthbar", it determine how much time you have left ingame. The mosquito will die when the energy hits 0%.
- Open-Closed Principle, a design pattern that says that a program should be easy to extend, but unable to modify.
- Highscore, a list containing the player's highest scores.

2 System design

2.1 Overview

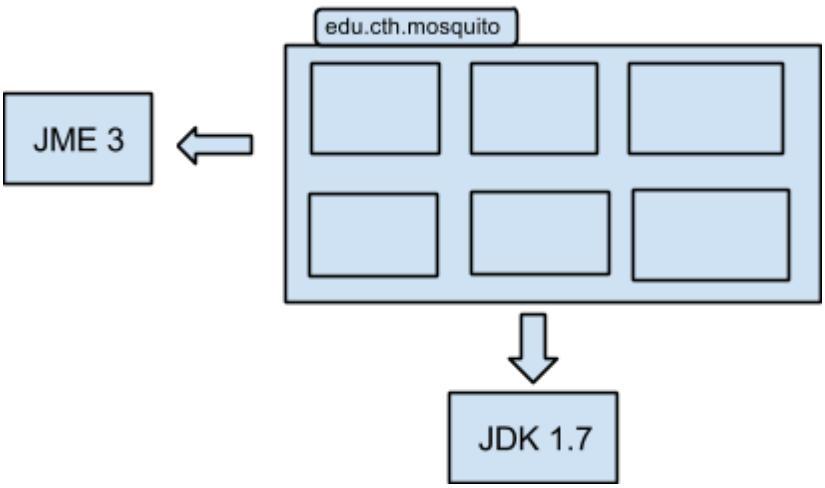
This application will use a modified MVC model.

2.1.1 Rules

The rules for the game are always the same and therefore we don't need a separate subsystem for this.

2.2 Software decomposition

Figure 1: This is an image of our games package structure. Where edu.cth.mosquito is the top level. It also shows that they are dependent on both JDK 1.7 and JME 3.



2.2.1 General

The application is decomposed into the following modules, see Figure 2.

- core, is the model of the game. Model part of MVC.
- view, main GUI for application. View part of MVC.
- controller, is the control classes for MVC model.
- filehandler, is for file handling.
- main, contains the main class holding the main-method, application entry point.
- util, contains utility classes that can be used as support for other classes.

Figure 2: The structure of the packages

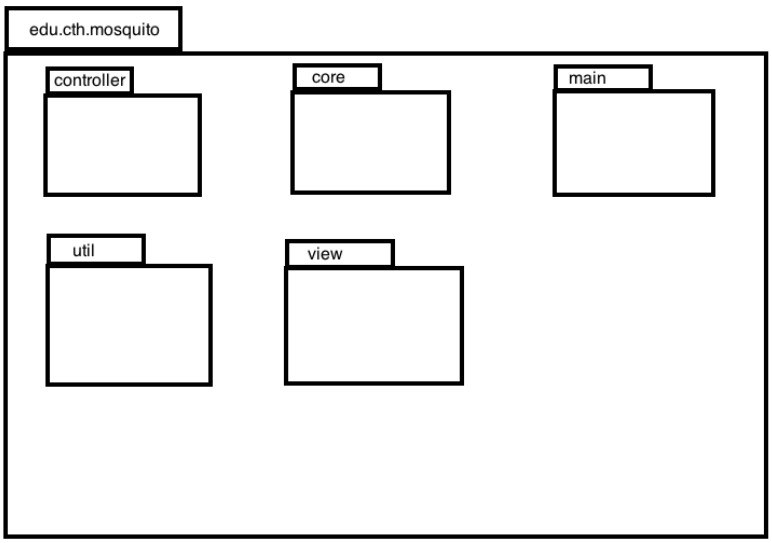


Figure 3. Classes within core package.

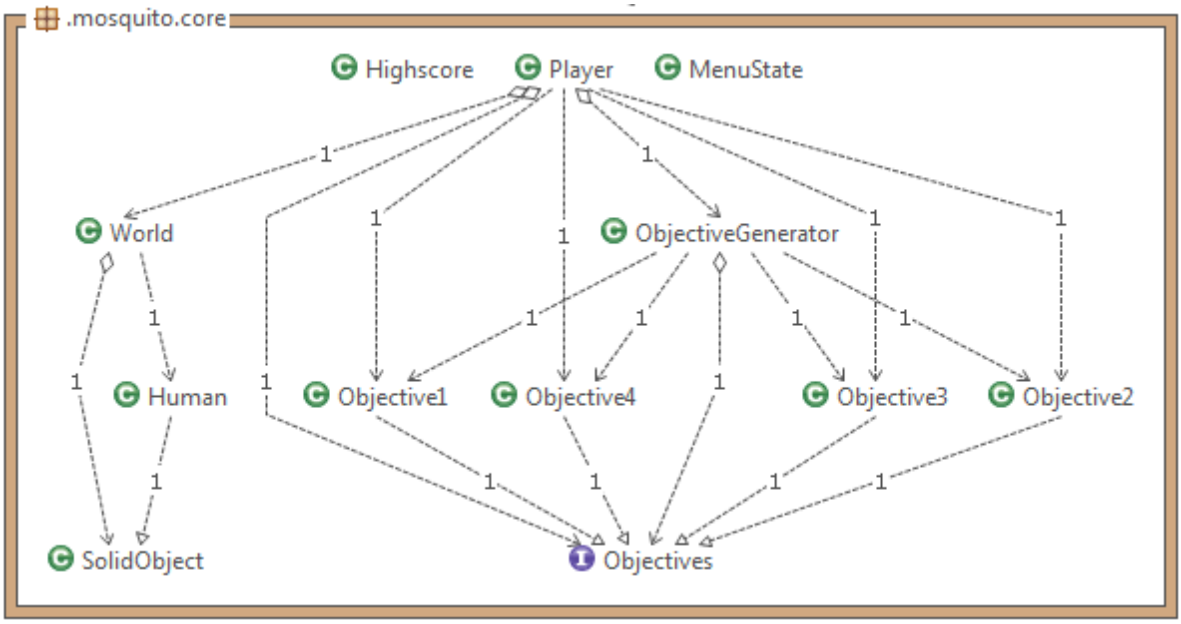


Figure 4. Classes within controller package

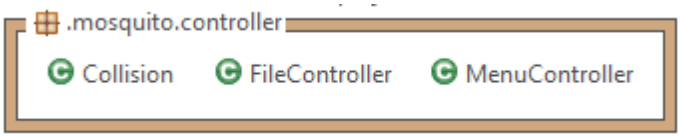


Figure 5. Classes within filehandler package

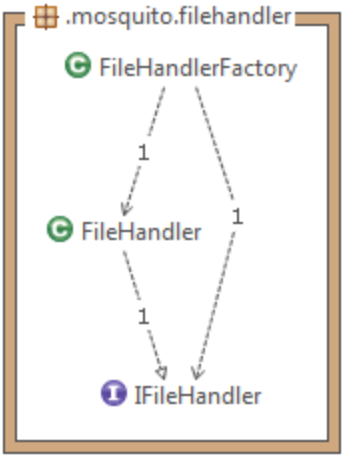


Figure 6. Classes within view package

The Dependency and layering analysis are shown in the chapters below.

2.2.2 Decomposition into subsystems

The only subsystem within this application is the filehandler class in the filehandler package. The classes in the subsystem only handles the filehandler package, so this is not a unified subsystem.

2.2.3 Layering

The layer is indicated in the image below.

2.2.4 Dependency analysis

Dependencies are shown in the image below.

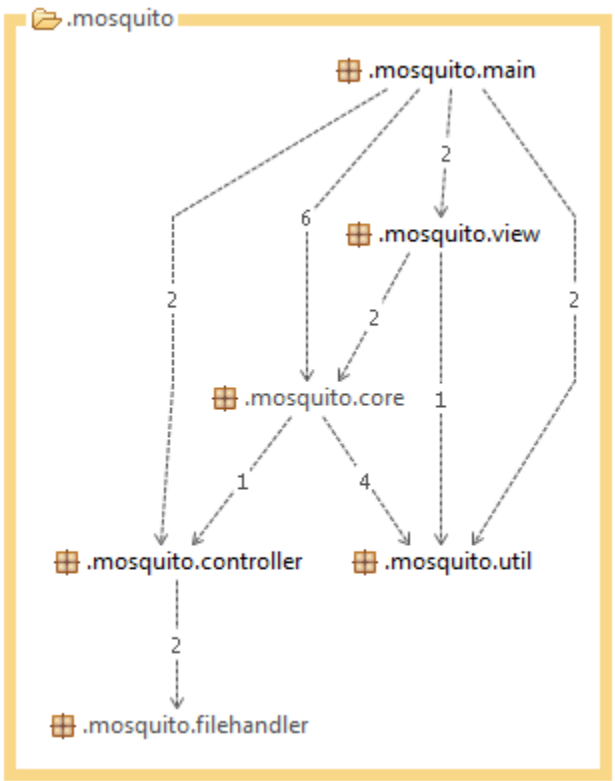


Figure 4: Dependency and Layering analysis.

2.3 Concurrency issues

This is a single threaded application. Everything will be handled by the jMonkeyEngines standard thread. Therefore no concurrency issues will occur.

2.4 Persistent data management

The only persistent data stored is for the highscore and this is stored in a .txt file.

2.5 Access control and security

NA.

2.6 Boundary conditions

Application launches and exits as normal desktop application as a jar file.

3 References

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