System design document for Group 8

Table of Contents

Version: 1

Date ⅖ -2013

Author Julia Friberg, Emma Westman, Oskar Dahlberg, Jonathan Thunberg

This version overrides all previous versions.

1 Introduction

1.1 Design goals

The model should be completely separated from the GUI so the GUI easily can be switched to another. It should be possible to isolate some parts of the model and test them. Save and read sevices for high score should be included and read service to read in waves as plain text.

1.2 Definitions, acronyms and abbreviations

- GUI, graphical user interface
- Java, platform independent programming language
- Monster, enemies trying to get to the end of the road.
- Towers, shot at the monster trying to prevent them from reaching the end of the road by shooting at them if the monsters are in range. Costs monye to get/upgrade towers.
- Road, monsters follow the road from the startpoint til the endpoint. If the monsters would reach the end the player loses a life.

- Players life, if the player loses all the lives it is game over.
- Wave, contains a group of monsters and make sure they spawn with a good interval.
- Level, one level contains a number of waves. If all waves passes and the player still has at least one life left the level has been successfully completed.
- Resources, the amount of money the player has to build towers with.
- 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.

2 System design

2.1 Overview

The application will use a MVC model with some GUI code in controller due to hard extraction in the Slick2D library.

2.2 Software decomposition

2.2.1 General

The application needs a computer with Java and Slick2D installed. JAVA 1.6 is needed for Mac users.

The main program is composed by the following modules:

- Main class includes the main-method that will launch the application.
- View, the GUI for the application. View part of MVC
- Model is the main model of the game. Model part of MVC.
- Controller is the controller part of the MVC, also contains some GUI code.
- The file package is responsible for saving and loading data.
- Sound handles the sound.

2.2.2 Decomposition into subsystems

The subsystems of the application can be seen in figure 2.2.2.1.

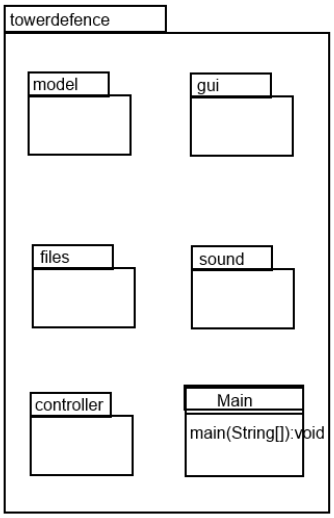


Figure 2.2.2.1

The main package is called towerdefence and in this package there is a main class that starts the game and five subpackages. An explanation of all the subpackages and their responsibility follows.

The model package includes all the data and logic to make the application work.

The gui package is responsible for how the application is shown on the screen.

The files package takes care of services regarding file handling.
Services that are included are save and read high score and a read function to determine which type of monsters to create in each wave.

Waves read from file
There is one file for each level. This file contains the type of monsters in all waves for that specific level. Numbers will represent the type of monster and each wave will be separated by a colon in the text file. The numbers used to represent monster types are 1,2 and 3. The number 1 will create a monster of the type "Monster", number 2 "MonsterFreezingImmune" and number 3 "MonserBurningImmune". So for example a file containing 12312:123:321 will send three waves. The first containing "Monster", "MonsterFreezingImmune", "MonsteBurningImmune", "Monster", "MonsterFreezingImmune" and so on for the next two following waves.

Save and read high score
In the model package there is an object representing a high score, a string for the name and and int for the points. Each level has a high score object and when the level has been successfully completed a method for saving the high score is called. This method first checks if there already is a high score registered for this level by looking for the levels name in the high score file. If the name exists it will compare this object to the old one, if the new one has a higher score the old high score object for the level will be removed from the file and the new one will be added, otherwise nothing will happen since the highest score already is saved. In those cases where there is no high score registered the score will be saved as a new high score. The high score is saved as an object using Serializable.

The sound package is responsible for all sound in the application. There is background music playing throughout the game and sound effects when the game is played. Both can be turned on and off during the game.

The controller package is split into two different parts. The first part is responsible for making sure the game works correctly.
The other part makes sure the right state is active. For example are there different states for the main menu and the actual game. This package also takes input from the player and tells the model to change according to what the player has clicked on.

2.2.3 Layering
The main class is in the highest layer. The next layer is the controller package which leads to all the other subpackages in the application. Under the controller are therefore the files package, the gui package, the sound package and the model package. The model package is a bit lower in the layering than the others since both the files package and the gui package have references to the model package. This is illustrated in figure 2.2.3.1.

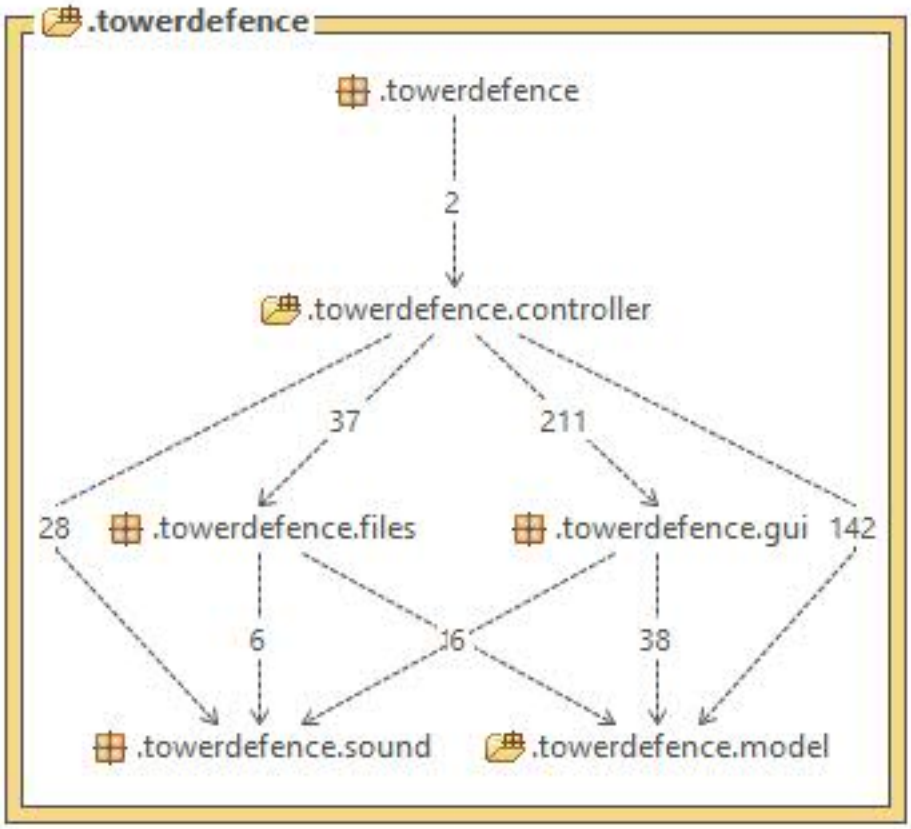


Figure 2.2.3.1

2.2.4 Dependency analysis
The applications dependencies are shown in figure 2.2.3.1. There are no circular dependencies in the application.

2.3 Concurrency issues
NA

2.4 Persistent data management
The high score for each level is saved in text files by the files package and therefore the progress as well. High scores are saved in a text file.

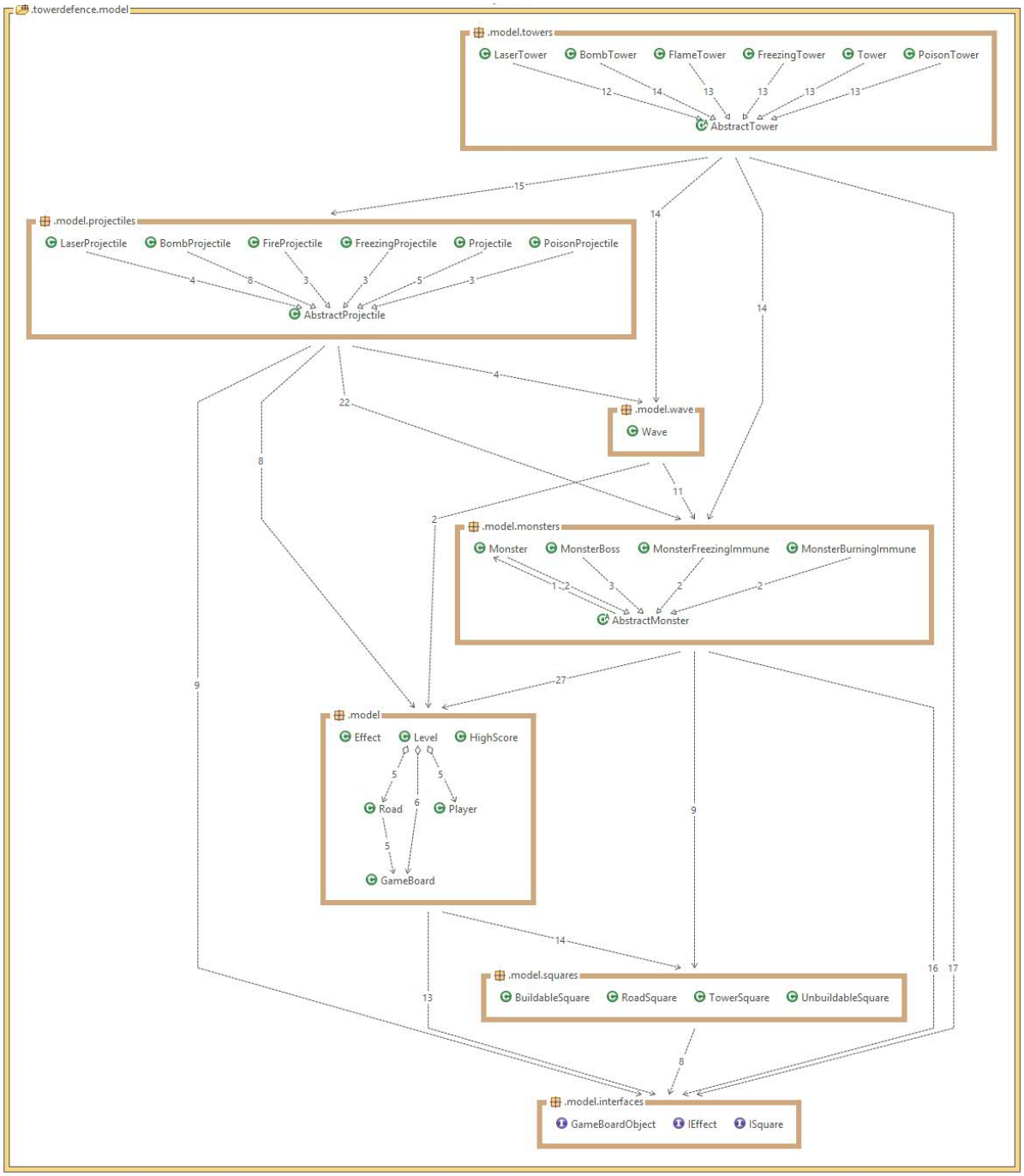
2.5 Access control and security

NA

2.6 Boundary conditions
NA

3 References

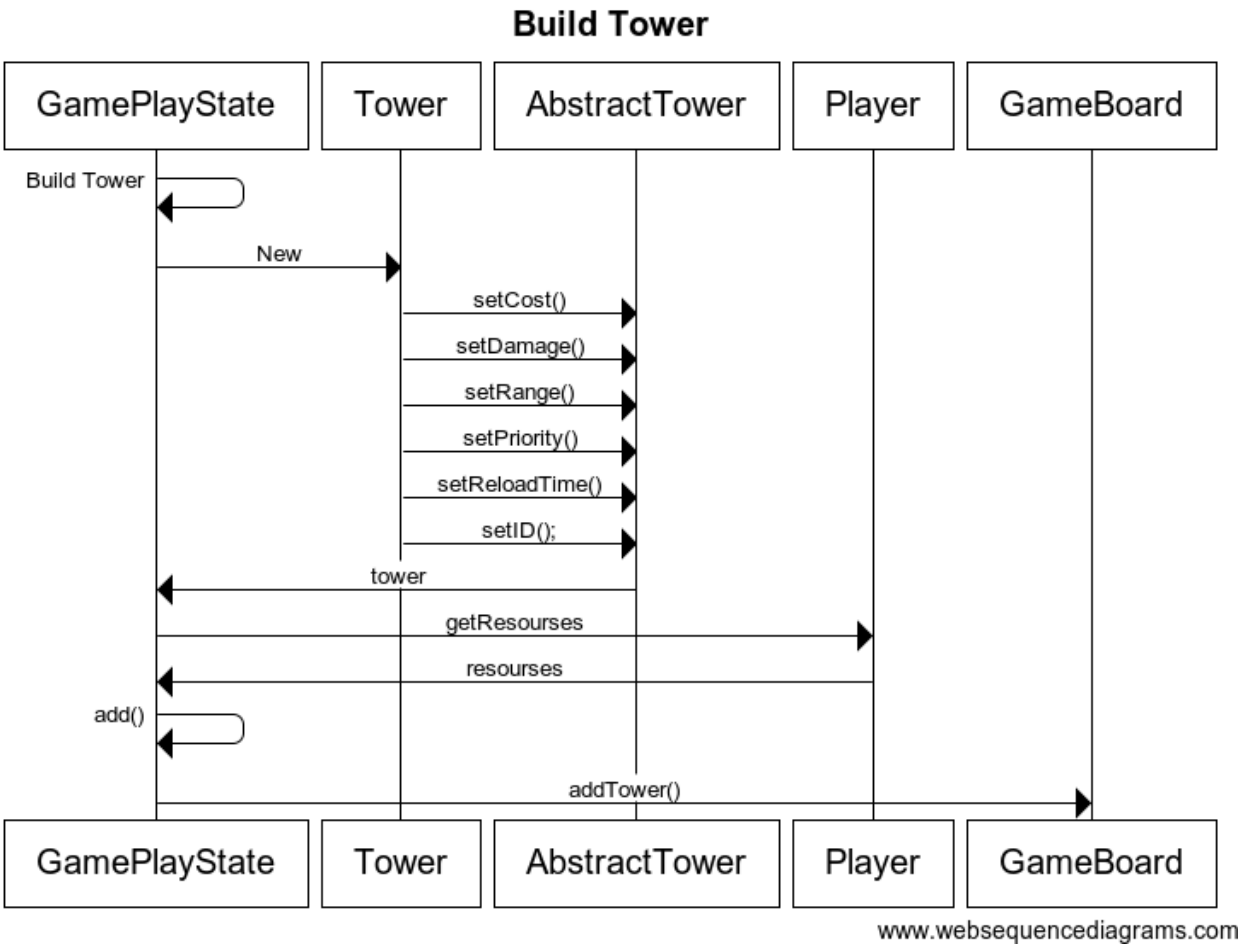
APPENDIX



- Level controls the functionality in the model.

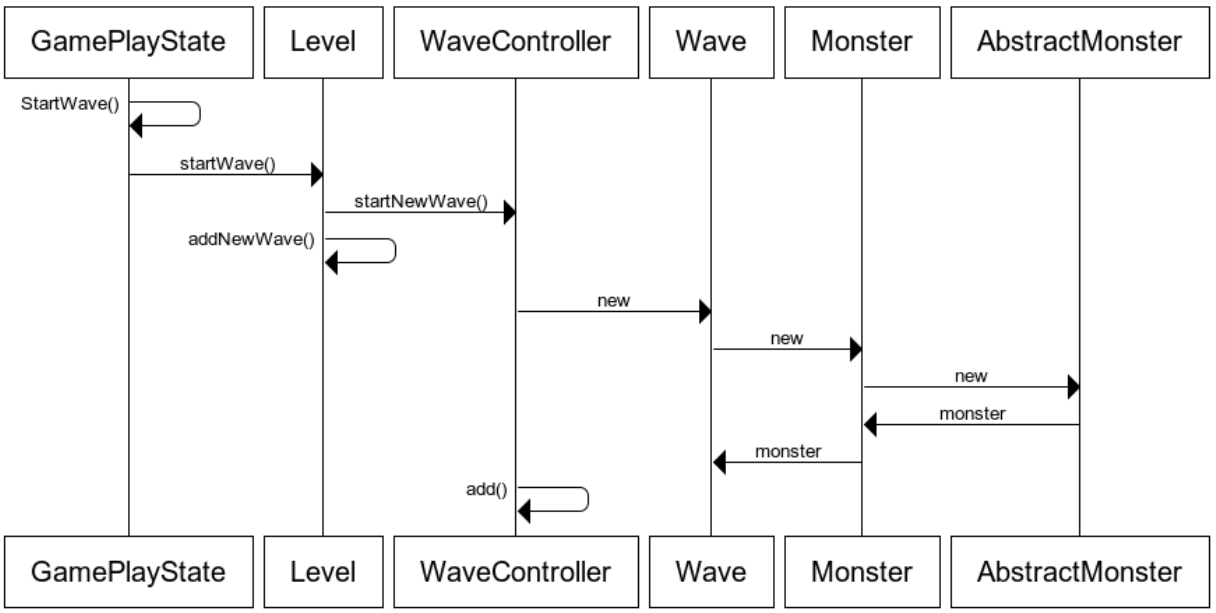
- GameBoard is a container for squares; ISquare, BuildableSquare, UnbuildableSquare and a Road.
- AbstractTower holds the main functionality for all types of towers.Tower, LaserTower and BombTower are subtyps of AbstractTower. All towers fires projectiles when a monster is in the towers range.
- AbstractMonster holds the main functionality for all types of Monsters. Every other kind of monsters are subtypes with some differences. All monsters moves individually and the logic for this is in the AbstractMonsterclass.
- Road is a container of RoadSquares.
- Player holds the the users lives, resources and points.
- A Wave is group of monsters. The Wave class holds all monsters in a wave.
- Projectile, LaserProjectile and BombProjectile are three different projectiles the tower can fire. These are all subtypes to AbstractProjectile. Projectiles can also have different Effects as freeze, burn and poison.

Sequence Diagrams



- HighScore is an object containing the highest point of a level.

Start Wave



www.websequencediagrams.com