CHALMERS

System Design Document for MachoDude, Grupp 13

Version: 1.0

Date: 25/5-2014

Authors	cid	mail
Alexander Alvmo	alvmo	alvmo@student.chalmers.se
Alexander Branzell	alebra	alebra@student.chalmers.se
Anders Stigsson	andstig	andstig@student.chalmers.se
Mikael Lönn	mlonn	mlonn@student.chalmers.se

CHALMERS

Table of Contents

1 Introduction

1.1 Design goals

The game logic and game design must be separated. All modules should be exchangeable and dependency should be kept low. The look and feel of the game is very important as is the responsiveness of the controls.

1.2 Definitions, acronyms and abbreviations

- LWJGL - Lightweight java game library.
- Slick2d - A game framework based on LWJGL.
- GUI - Graphical User interface
- Java - Programming language
- Player - The person playing the game
- Tile - A small image containing specific properties
- Class - A file containing code
- Abstract Class - A Class which other classes work towards

2 System design.

2.1 Overview

The application will implement MVC and run the slick2d game engine. The levels will be created using the Tiled while the tiles and all entities will be created in Adobe Photoshop. All interactive objects will implement the abstract entity class. Several interfaces are to be created, such as IMoveable for entities that are supposed to be able to move like the player or enemies. The design focuses on a number of states that are interchangeable, each state holding a different view to the player. These are controlled by the GameController class.

2.1.1 Entities

Every renderable object in the game is an entity. An entity contains basic information regarding the objects position and how it should look like, that is all. The object itself contains most of the more specific logic on exactly how it interacts with other objects in the game.

2.1.2 Level

Slick2D supports the .tmx (Tilemap) file format that Tiled uses. The GameModel loads the specific Level. The level itself iterates the tilemap and interprets the different tiles based on their information. It then creates the specific objects for each tile and places them at the correct position, thus building the level. The model then moves the entities based on the players movements to create a scrolling effect.

2.1.3 2.1.3 States

There are seven different states, excluding Slick2D's BasicGameState. AbstractMachoDudeState is the abstract class which all the other states (except GameState) extend. It sets the basics like the background image and a menu. The MenuState class contains the menu where the player navigates to some of the other states. The options are:

- Start Game - Which sends the player to the LevelState where the player can choose which level to play, it then sends the user into GameState where the gameplay begins.
- Settings - Where the player can select the controls for the game in SettingsState
- Levels - Where the player can select what level to play.
- Stats - StatsState displays information about the player's statistics.
- Quit - Exits the game and prompts the player to keep playing.

If the player should die, he is sent into the GameOverState where score, high score and a navigation to the main menu is shown. There is also a WinningState if the player completes a level. It is very similar to the GameOverState, the only differences is that the text that says GameOver is changed to Congratulations and the user can go directly to LevelState.

2.2 Software decomposition

2.2.1 General

Most classes in the application works towards an interface, as the UML diagram shows no classes have circular dependencies. The modules used are:

- Game - Holding the main method for starting the application
- Controller - Contains the different states and the controller for the states
- Entities - All the entities for the game are placed here

- Exception - The java exceptions used for error handling
- Level - This is the where the levels and the level abstract class is stored
- Model - The game Logic
- Particles - Code for creating all the particles used in the game
- Util - Utilities such as cameras and constants
- View - Containing code for the different views that are updated by the model

2.2.2 Decomposition into subsystems

The Slick2D engine is a subsystem to the application.

2.2.3 Layering

Each component can be reused and reimplemented as can be seen in the appended diagram. The higher up in the image the higher up in the layering hierarchy.

2.2.4 Dependency analysis

There are no circular dependencies, as seen in appendix:

2.3 Concurrency issues

The Slick2D engine runs four simultaneous threads, but other than that the game itself is unthreaded. Should concurrency issues show then the fault is within the engine and not the application.

2.4 Persistent data management

Level progress and statistics are saved in the file “res/users/stats/stats.save”.

The integers representing the controls are saved in the file “res/user/Controls/controls.save”.

The tmx-files for the levels are saved located in “res/Maps/”.

2.5 Access control and security

N/A

2.6 Boundary conditions

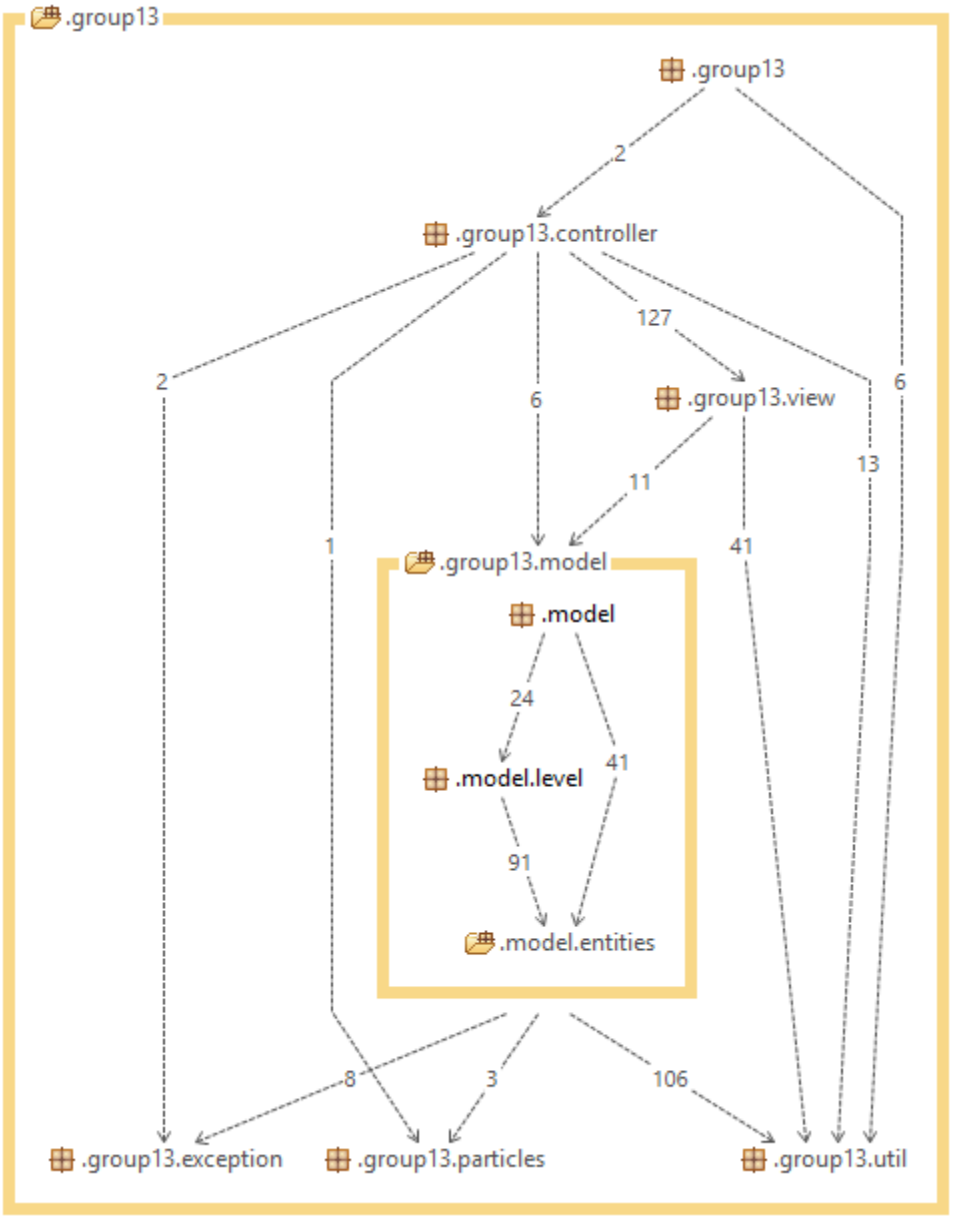
Application is started as a normal desktop application and can exited from the in-game menu.

3 References

-

APPENDIX

Layering and Dependency analysis



CHALMERS

Package diagram