System design document for Twirl

Table of Contents

Version: 0.1

Date 19/4 - 2013

Author Eric Arnebäck

This version overrides all previous versions.

1 Introduction

1.1 Design goals

The design should use MVC in a way that makes it easy to reuse the model when porting the game to other platforms. The physics engine should also be easily replacable. The model should be testable.

1.2 Definitions, acronyms and abbreviations

(TODO: write definitions of terms here that we use throughout the document)

2 System design

2.1 Overview

The game uses a MVC model.
(TOOD: describe the model)

2.1.1 Entities

All the ancestor of all the entities is the abstract Entity class. But no entity used in the game directly extends Entity, they rather extends one the classes CircleEntity, PolygonEntity, or RectangleEntity. Which one they chose to extend depends on their shapes. I.e., Obstacles are polygon-formed so they extend PolygonEntity.

2.1.2 Level

2.1.3 Universe

2.2 Software decomposition

(this section basically describes the package structure of the java application)

2.2.1 General

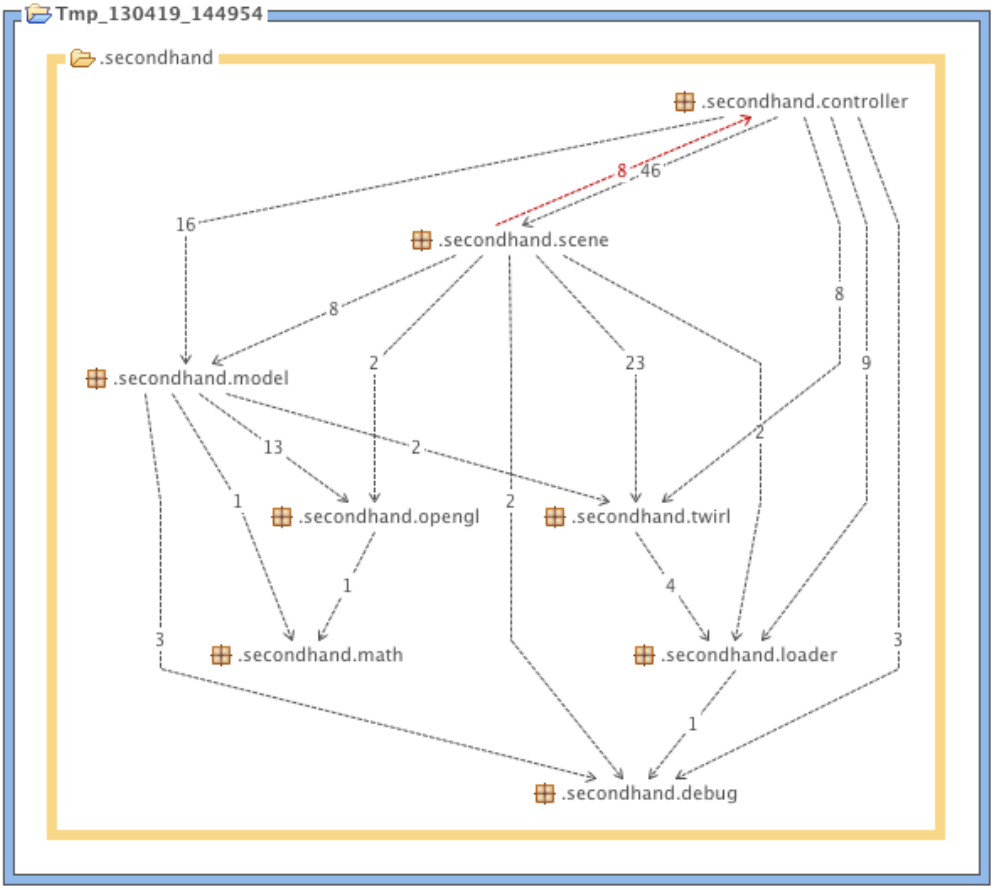
- controller, the controller classes of the MVC.
- debug, debugging and logging.
- loader, singelton classes that simplify the loading of resources in AndEngine
- math, math related functionality, there's for example a triangulator in here.
- model, the model classes of the MVC.
- opengl, graphics related functionality that we needed but AndEngine didn't directly support. AndEngine for example has no support for drawing circles or polygons, so we had to write classes to extend AndEngine to support this. We used OpenGL to draw the graphics, hence the name.
- Scene, every screen in the app is a called a scene in AndEngine(ie, the main menu screen is one scene). All the scenes of the app is stored in this package.
- Twirl, ??? What do we put here?

Package diagram. For each package an UML class diagram in
appendix

2.2.2 Decomposition into subsystems

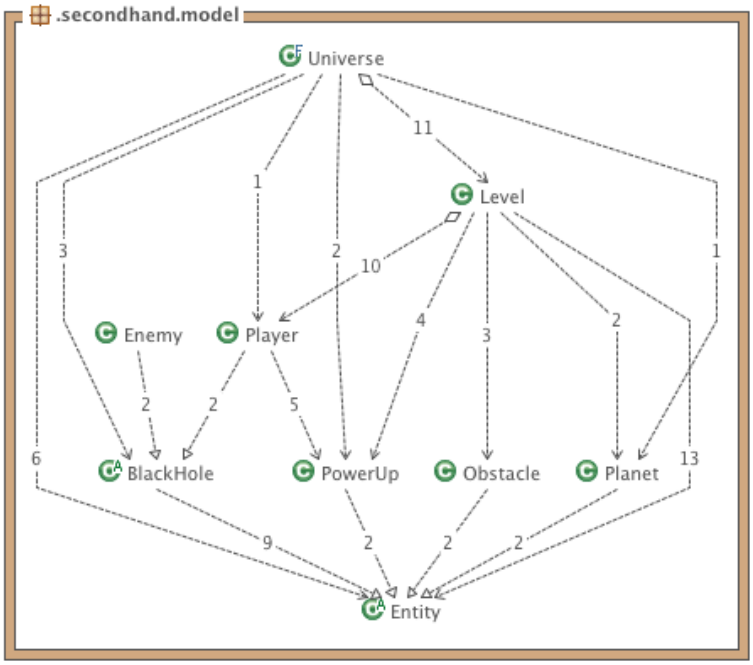
Don' think we have any subsystems. Everything is handled in one large system.

2.2.3 Layering
The layering is as indicated in the Figure below . Higher layers are at the top of the figure.
2.2.4 Dependency analysis
Dependencies are as shown in Figure. There are no circular dependencies.
(TODO: insert a Stan generated diagram below to show the laying and dependencies.)
2.3 Concurrency issues
The only multithreaded part of the application is the loading screen, right? That thing is shown for such a short time that it really shouldn't cause any problems.



(the above create image,, right click the project and press Run As > Structure Analysis, then right click on the diagram and export as png.)
2.4 Persistent data management
All persistent data will be stored as XML. The files will be:

- file for storing high scores.
- Localization files for the localizible text of the app.
- Probably ones for storing the levels as well.
- The settings of the app.



2.5 Access control and security
NA(I don't think this is something we'll have to worry about)
2.6 Boundary conditions
NA. The app is launched like any other android app.
3 References
(TODO: Cite important design sources here. Ie, mvc. Not anything related to code details!)

APPENDIX

(TODO: we should put a class diagram for all the packages.can easily be generated using stan)