System Design Document for the RTS survival project.

Contents

Version: 1.0

Date: 2012-05-21

Author: Filip Brynfors, Markus Ekström, Björn Persson Mattsson, Jakob Svensson

1 Introduction

1.1 Design goals

The design should be loosely coupled and testable.

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 a Java application.

Host, a computer where the game will run.

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 us a MVC model.

2.1.1 Rules

NA

2.1.2 The model functionality

The models functionality will be exposed by the interface IGame.

2.1.3 Event handling

The GameView listens directly to abiliies and EntityManager.

2.1.4 Internal representation of text

All texts are in english only.

2.2 Software decomposition

The application is decomposed into the following modules, see Appendix.

- controller, contains all classes that handle interaction between the user and the application.
- view, contains everything in the scene (e.g spatials).
- model, is the top level package for all model related classes, including the game core.
- model, contains everything that has anything to do with the game model.
- Main is the application entry class.
- io, contains the loading of files.

2.2.1 Decomposition into subsystems

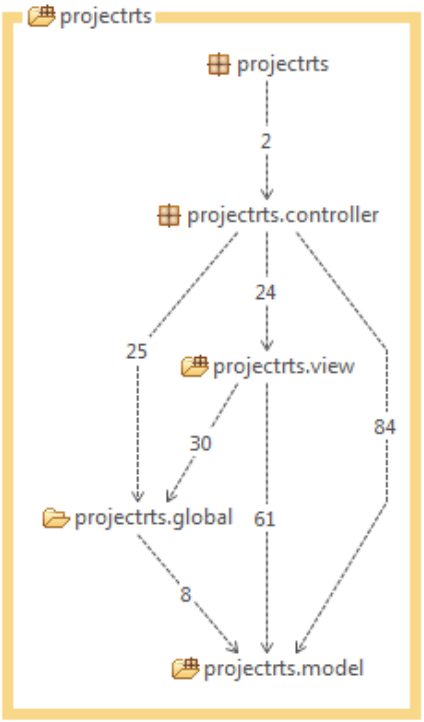
The application uses the JMonkeyEngine 3 framework. Depending on the definitions used, this framework can be seen as a subsystem. The application also uses an input/output-package which may also be seen as a subsystem. The Nifty framework is used for for the GUI in the application.

2.2.2 Layering

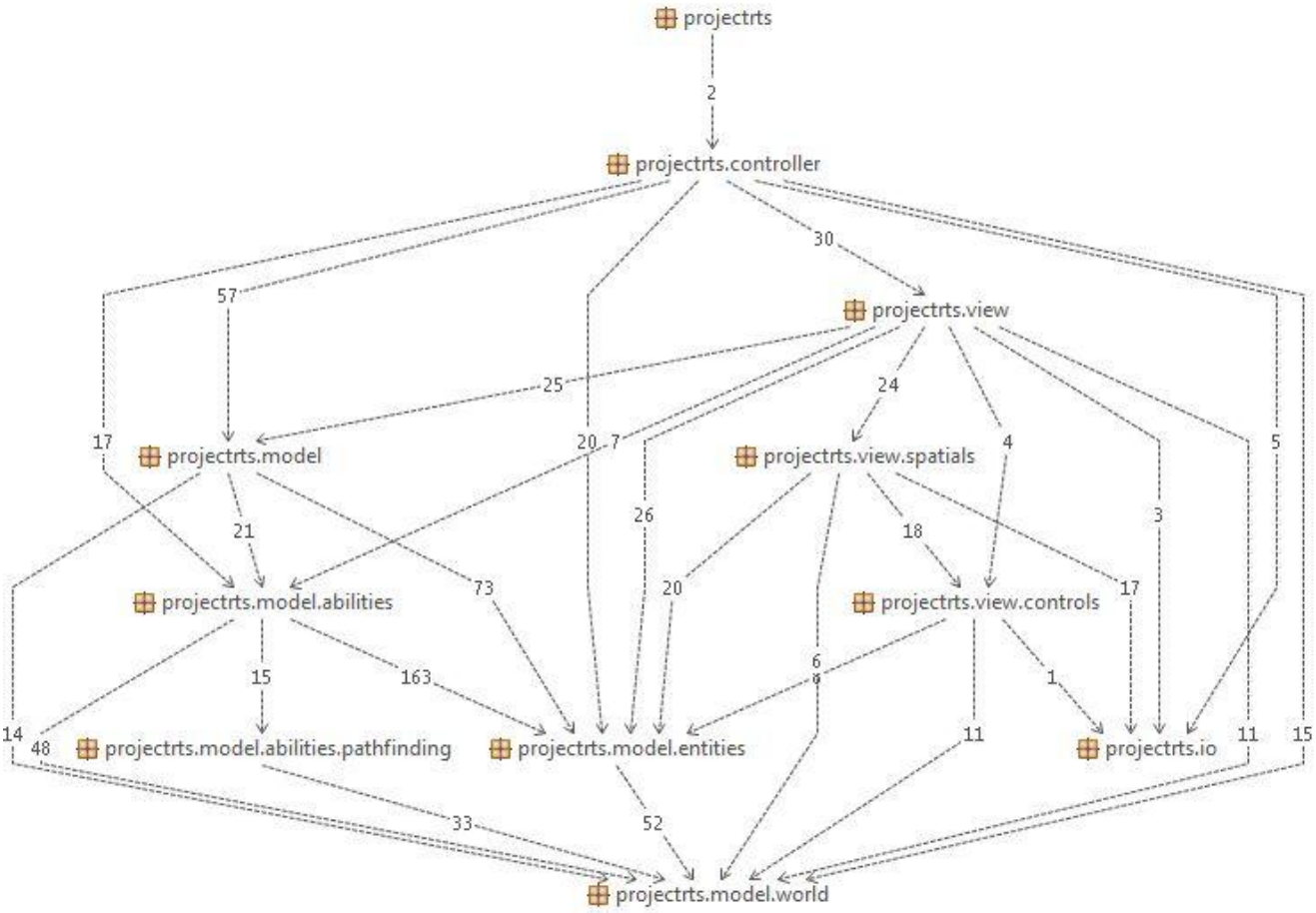
The application is divided into three layers, control layer, view layer and domain layer

2.2.3 Dependency Analysis

Top-level



Flat package



2.3 Concurrency issues

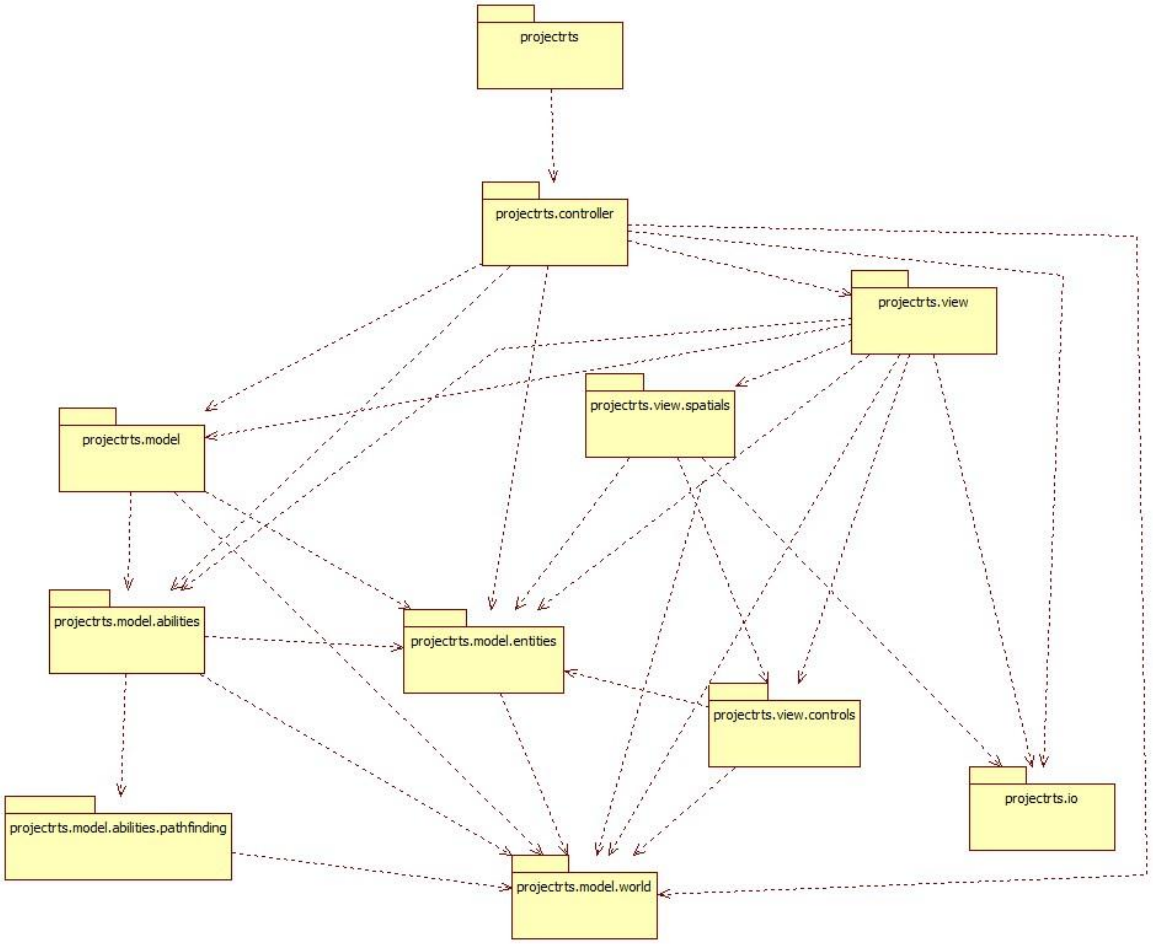
Since the A* algorithm is running in another Java thread there could be some concurrency issues. The main problem that could appear is that two adjacent units would ask for a path at basically the same time. That would possibly lead to a problem where both units consider the path's first node to be walkable, and therefore both units would be able to occupy the node at the same time. However, this will not arise often enough to be a problem and it will have no significant impact on the game play at all.

2.4 Boundary conditions

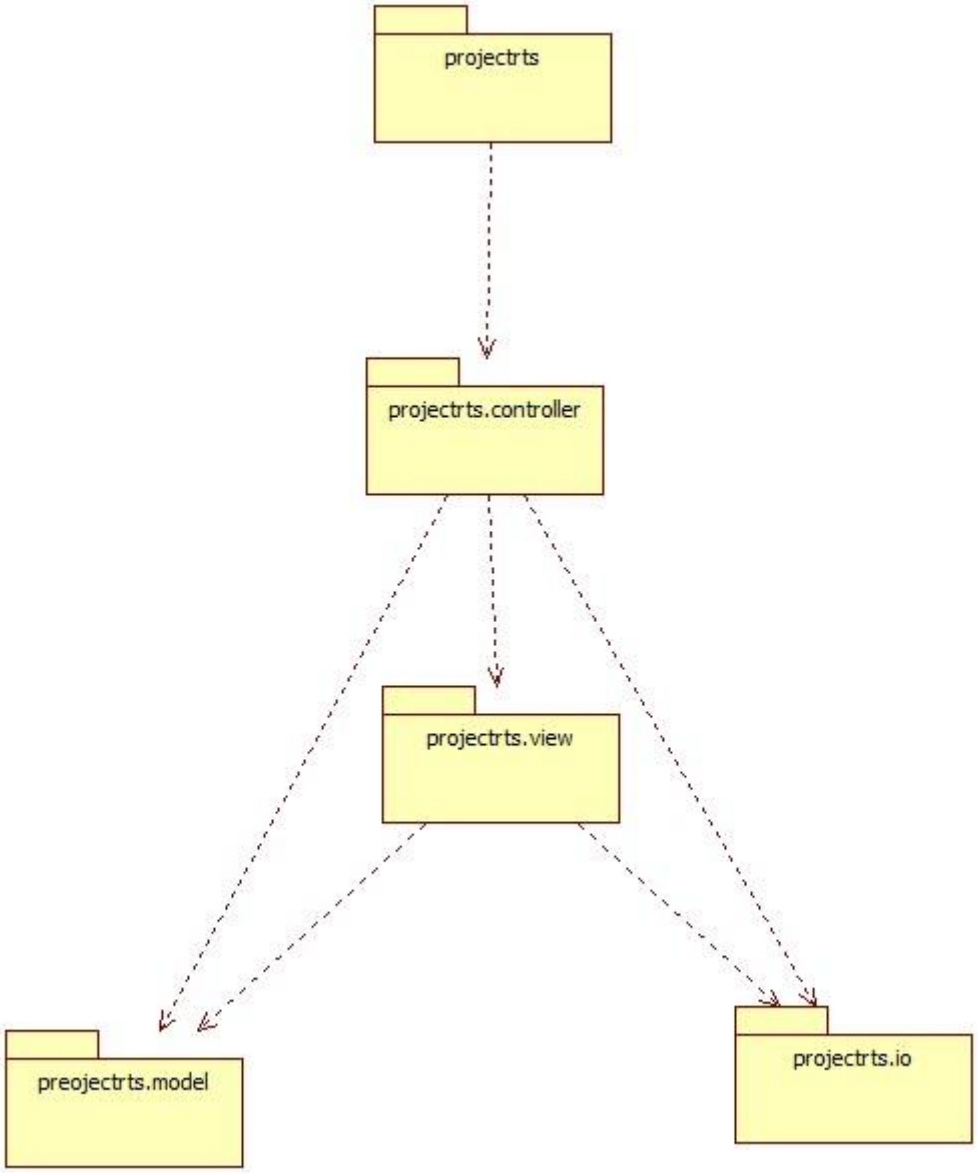
NA

Appendix

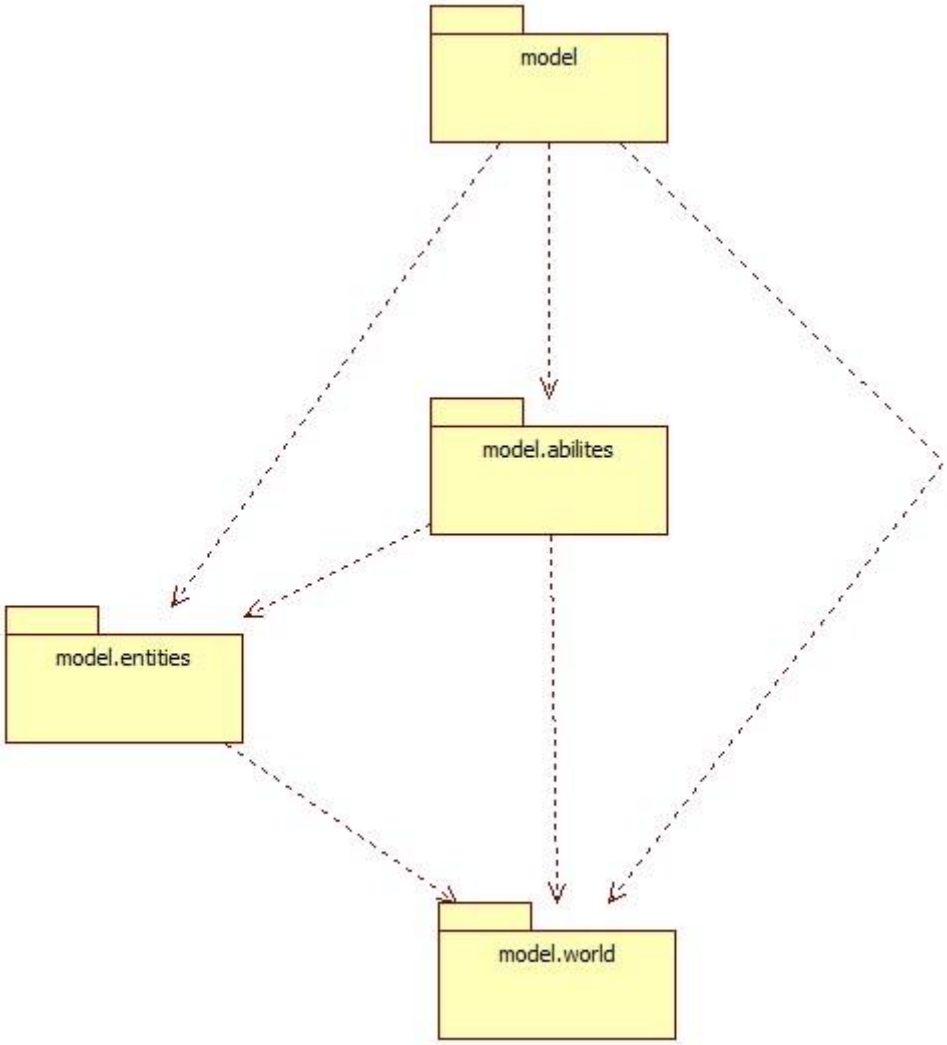
Top level



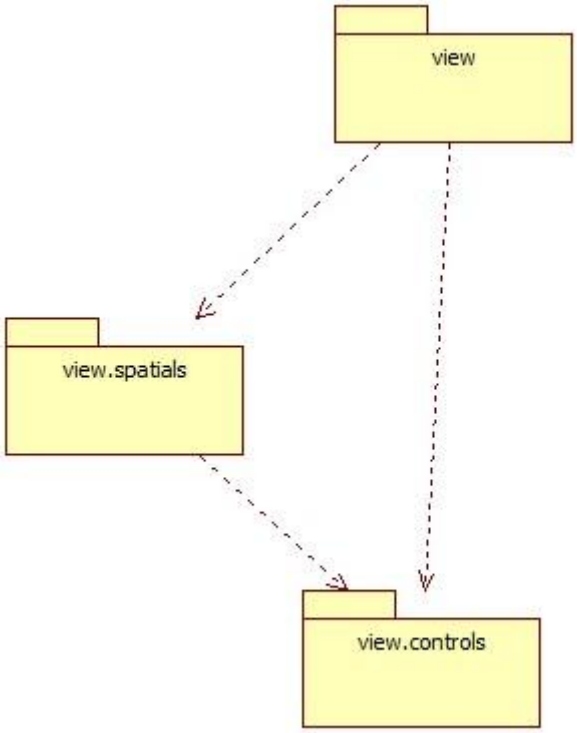
Simplified top level



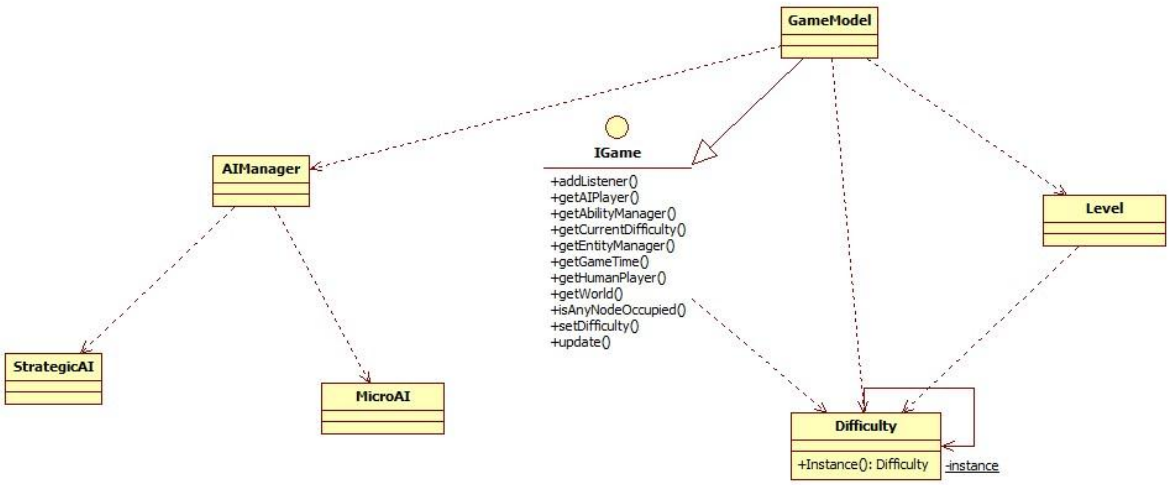
Model



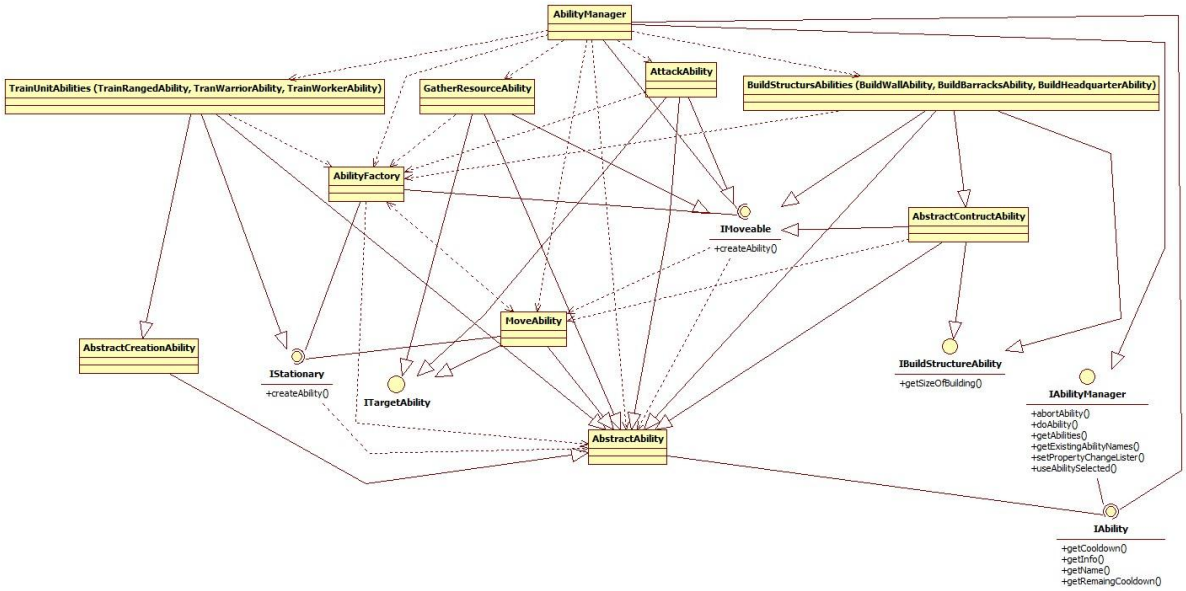
View



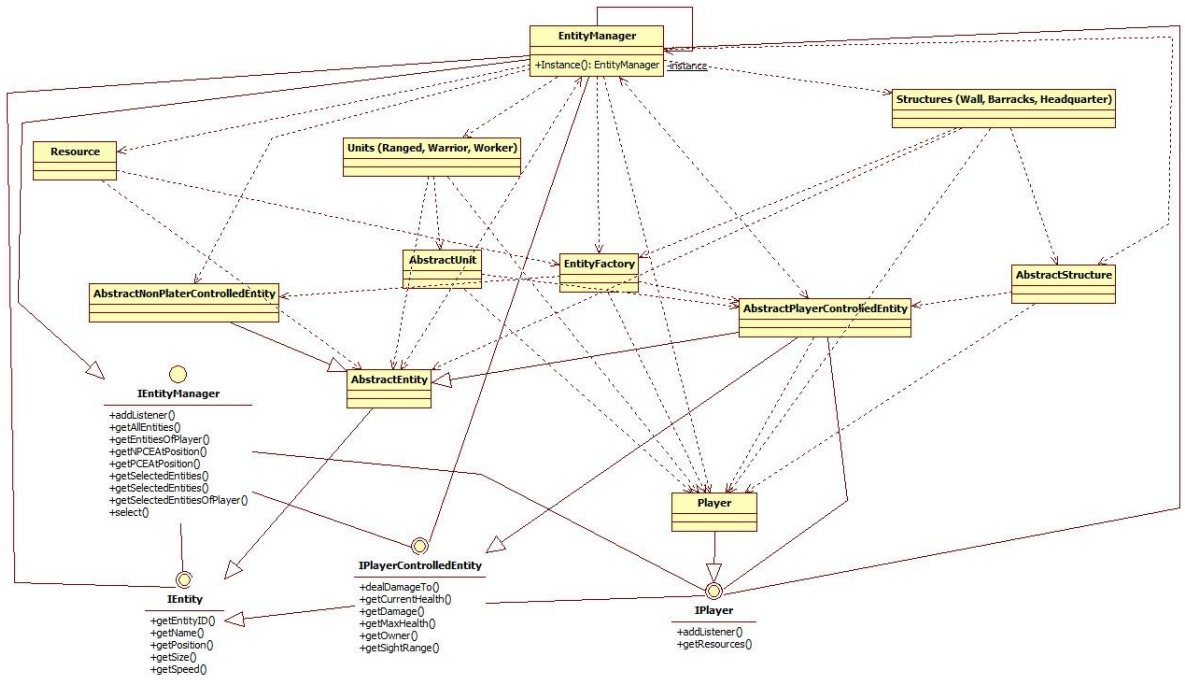
Model without sub packages



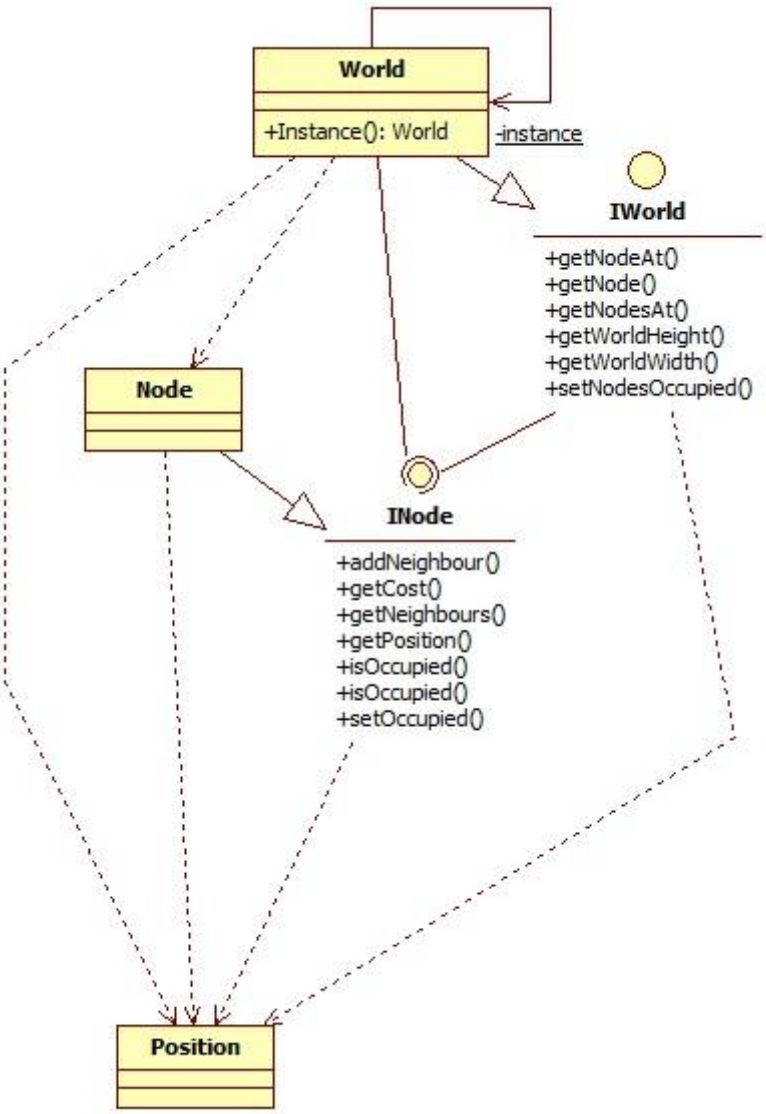
Abilities



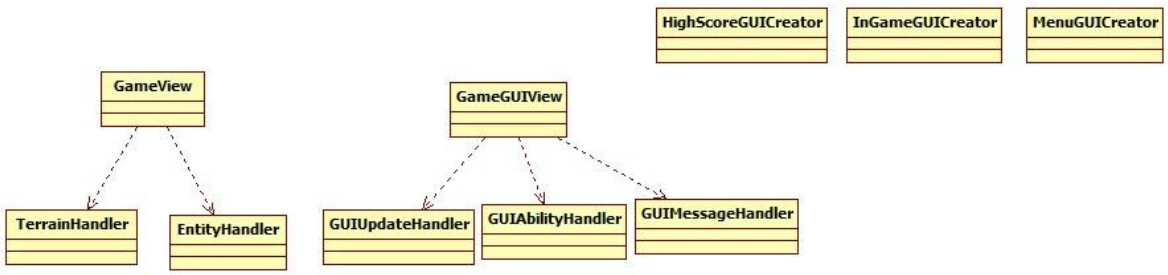
Entities



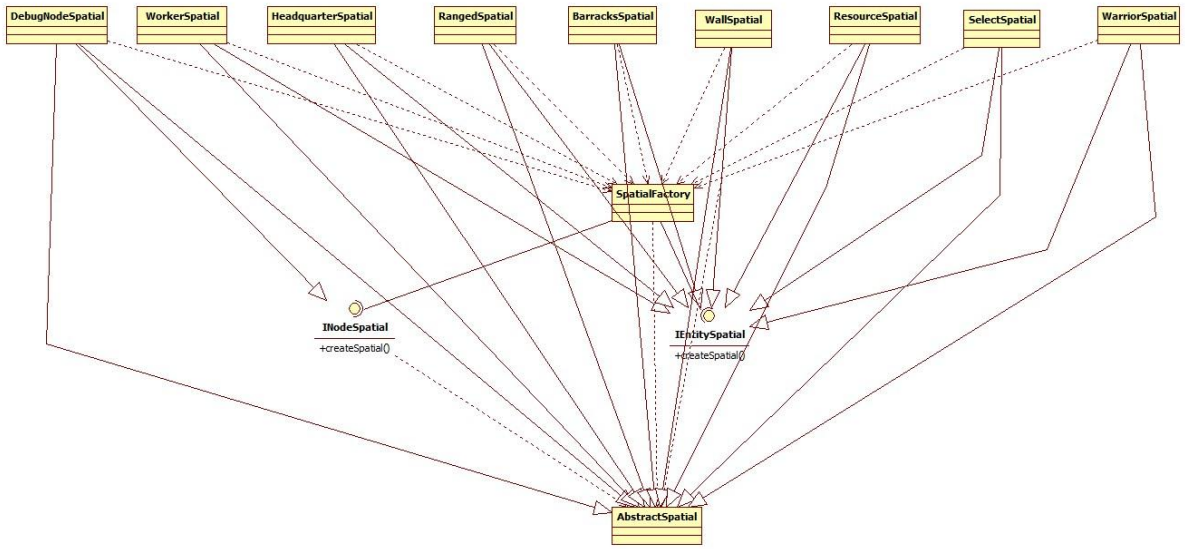
World



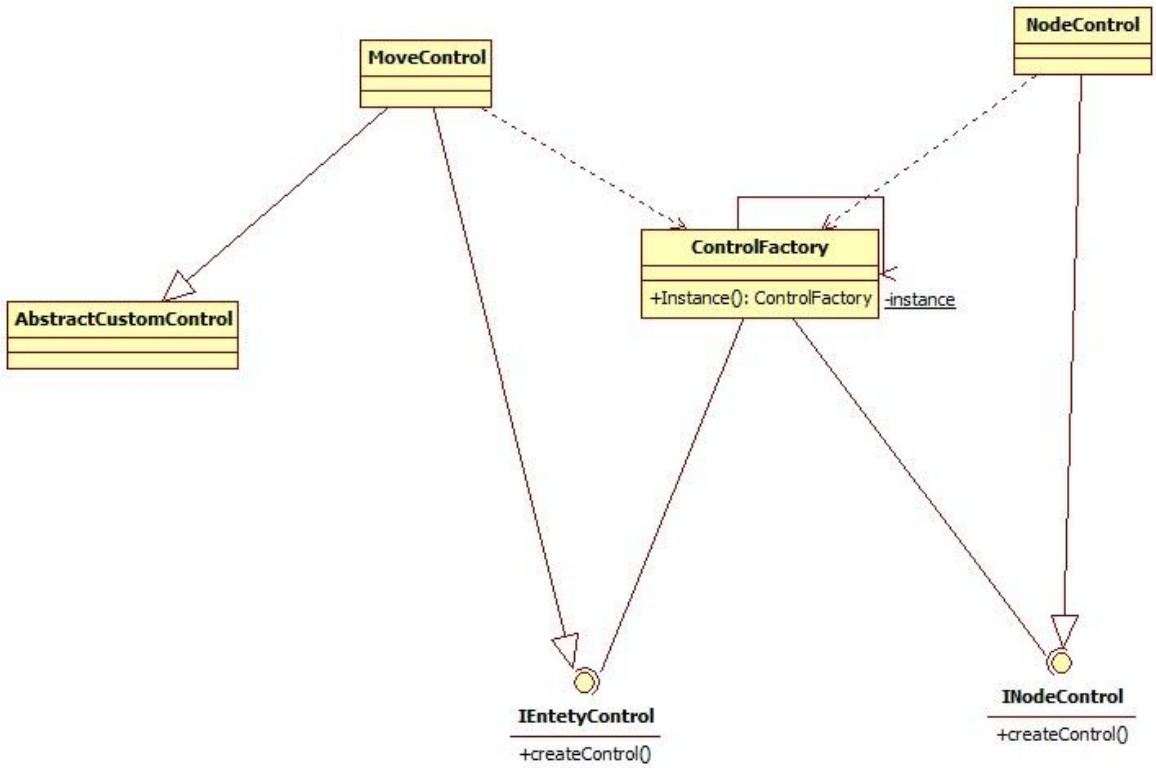
View without sub packages



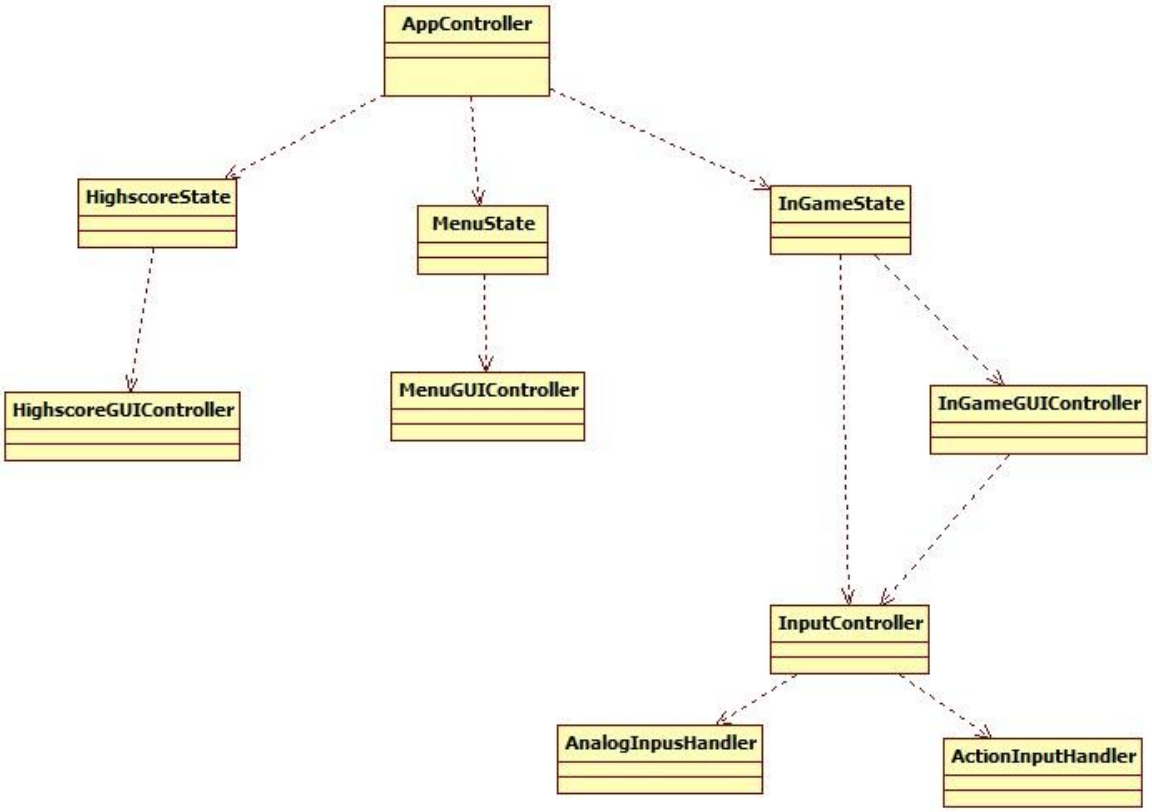
Spatials



Controls



Controller


AppController
HighscoreState
MenuState
InGameState
HighscoreGUIController
MenuGUIController
InGameGUIController
InputController
AnalogInpusHandler
ActionInputHandler