SYSTEM DESIGN DOCUMENT FOR VOXEL GALAXY: ARENA

Version:1.0
Date:29/05-16
Author: Simon Mare, David Iliefski Janols, Hannes Thell, Lukas Wallner

This version overrides all previous versions.

CONTENTS

1 Introduction

1.1 Design goals

The design must follow a clear MVC-pattern, to facilitate eventual switches of GUI and game engine. This means that the model will only contain the game logic and no graphics whatsoever, thus becoming completely independent from the view and the controller. The design must be testable, which means that it should be possible to isolate modules and classes for test. For usability see RAD.

1.2 Definitions, acronyms, and abbreviations

- GUI, graphical user interface
- Java, platform independent programming language.
- MVC, Model-View-Controller.
- jME3, jMonkeyEngine 3, a game engine.

2 System design

2.1 Overview

The application will use an MVC model based on the jME3 framework. The control layer will call the model for information every update frame and use it to update the graphics via the view layer.

2.1.1 Initializing the game

jMonkeyEngine applications are not made to be initialized with an MVC model in mind. Therefore, we have made two classes, viewInit and controlInit, to separate the view from the controller during the initialization. Since jMonkey requires at least one class to extend its SimpleApplication-class, which contains almost all game assets, such as the input and asset managers, the root node etc., and all other classes needs access to this class, this class has to be initialized first.

2.1.2 The model functionality

The World class acts as a holding class for the game world, having the other model classes as field variables (one instance of Terrain, and two instances of Player). It also holds the data required by the view classes to create and control the game (the dimensions of the world and the model of the camera), as well as the possibility to create new instances of the PowerUp classes to be spawned in the world by the controllers.

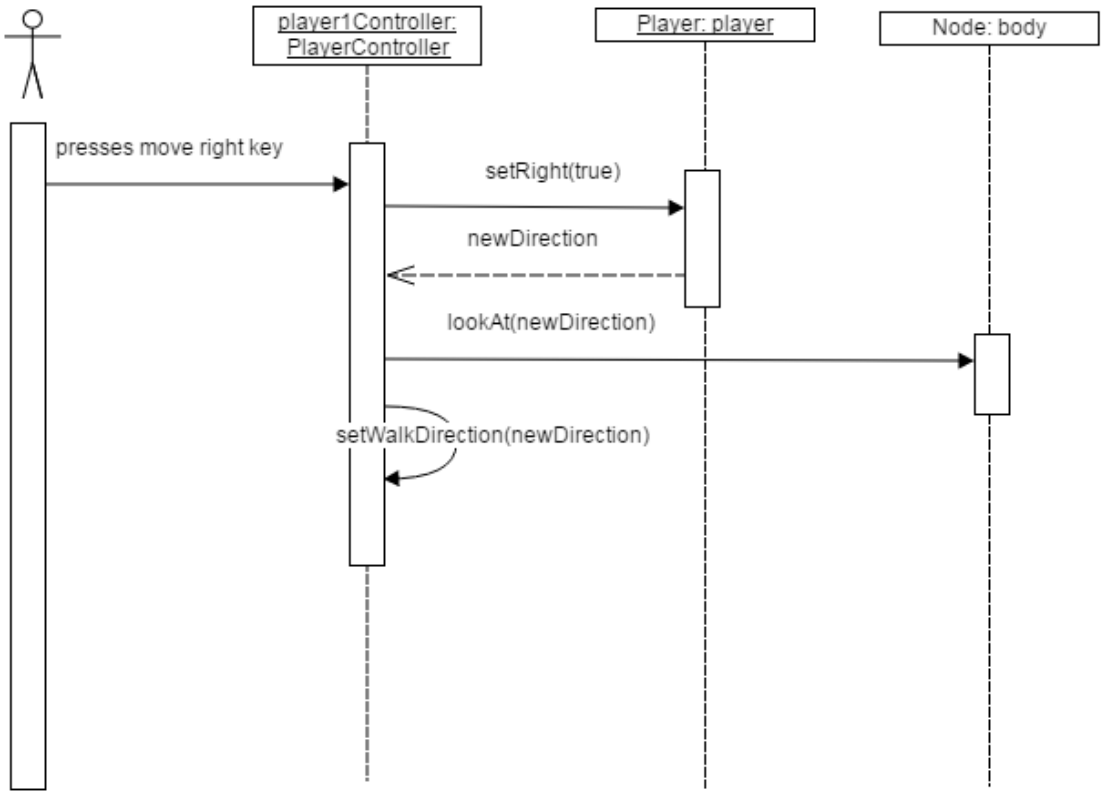
2.1.3 Event paths

User input will be handled by the event handling mechanisms of jME3 (inputManager). Call chains are like: Controller classes -> Model classes, Controller classes -> View classes.

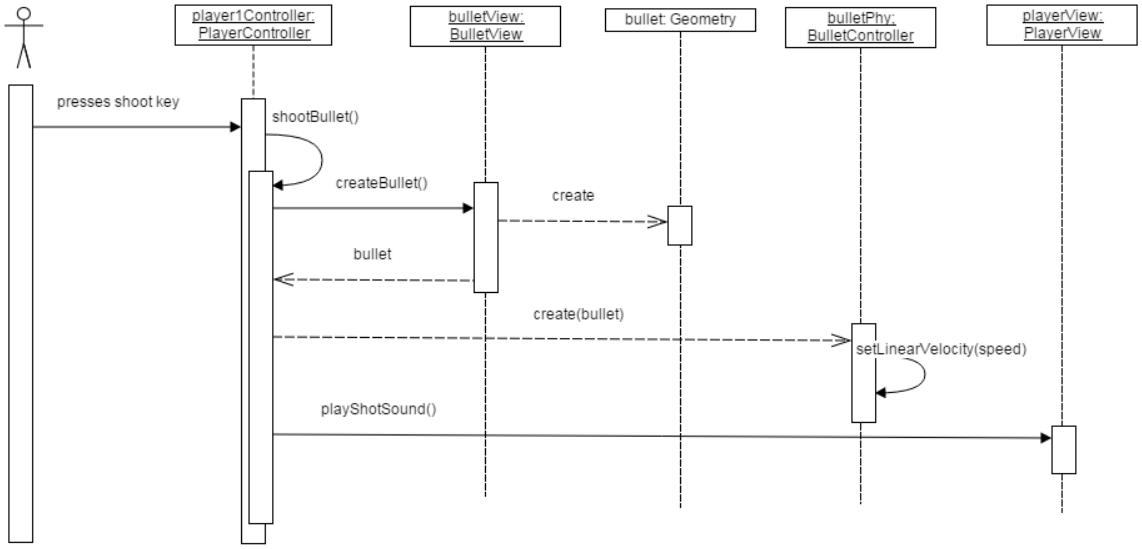
The controller converts the user inputs to information that can be sent to the model, which in turn uses the information to calculate and update its state. In the same update frame, the controller reads the state of the model and uses it to update the view. All updates of the model will be done by setter methods, and the controller will read them via getters.

2.1.5 SEQUENCE DIAGRAMS

move()



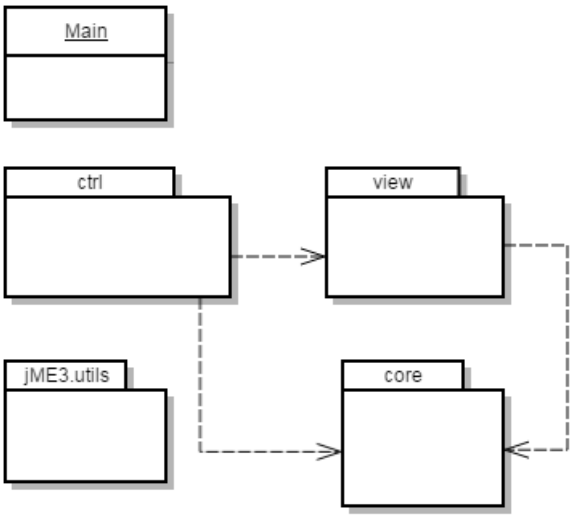
shootBullet()



2.2 SOFTWARE DECOMPOSITION

2.2.1 GENERAL

The application is decomposed into the following top level packages (arrows for dependencies)



- Main is the application entry class
- View is the top level package for all GUI related classes
- Core, the OO-model
- Ctrl are the use case controllers
- jME3.utils are general utilities related to the framework

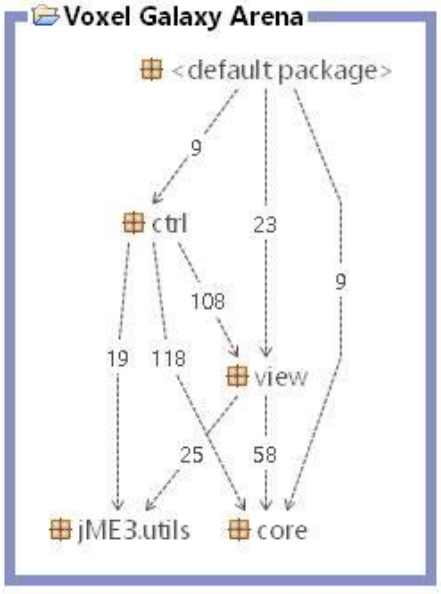
2.2.2 DECOMPOSITIONS INTO SUBSYSTEMS

NA

2.2.3 LAYERING

NA

2.2.4 DEPENDENCY ANALYSIS



2.3 Concurrency issues

NA

2.4 Persistent data management

NA

2.5 Access control and security

NA

2.6 Boundary conditions

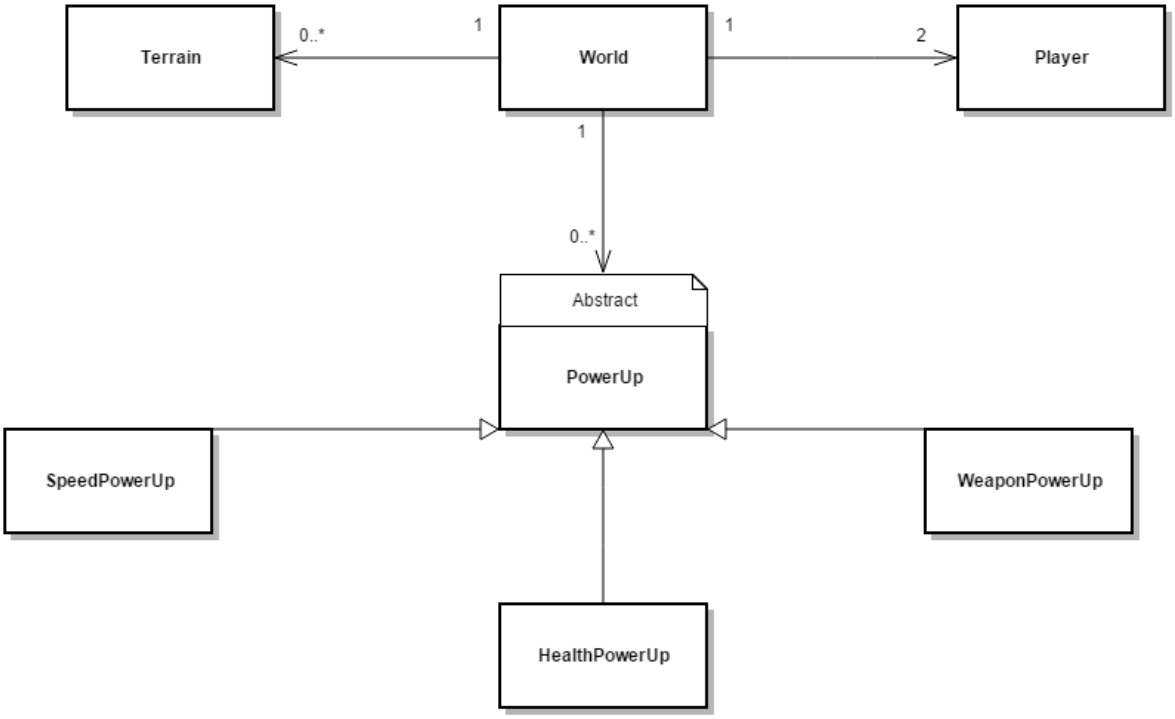
NA

3. References

MVC, see http://en.wikipedia.org/wiki/Modelviewcontroller

Appendix

Designmodel



Softwaredependencies
jMonkeyEngine3
jUnit
Nifty