Software Design Document

Group 16

Version: 1.0

Date: 31/5-15

This version overrides all previous versions.

1 Introduction

1.1 Design goals

Our goals with this project is to create a 2D platforming-game using the game-engine LibGDX. The game shall follow the Model-View-Control design pattern.

1.2 Definitions, acronyms and abbreviations

Active object - Active or interactive objects which exist within the game but are impossible to pick up, such as light switches or movable crates.
FPS - frames per second
JRE - the Java Run time Environment. Additional software needed to run an Java application.
GUI - Graphical User Interface
Viewport - Screen space in which the game world is drawn, as opposed to interface controls.

2 System design

2.1 Overview

The project follows the Model-View-Control design-pattern.

2.2 Software decomposition

2.2.1 General

See Appendix for Package diagram.

2.2.2 Decomposition into subsystems

The game has a couple of subsystems. One subsystem is the Flashlight which calculates a polygon which should represent the area the flashlight shines on. Another subsystem is the collisiondetection. This system calculates whether the player collides with a wall. If so then it moves the player out of the wall.

2.2.3 Layering
N/A

2.2.4 Dependency analysis
The only used dependency is LibGDX. This is a game engine able to create games for many different platforms. In this project this dependency is used to help draw the games different levels as well as handle inputs. References to the dependency can as such only be found in ProjectController and in ProjectView.

2.3 Concurrency issues
Concurrency is not an issue for this application. The application uses LibGDX to handle inputs and it only updates inputs once every main-loop. As such values set by the inputs should stay the same during the whole main-loop.

2.4 Persistent data management
Texture files are saved as .pack-files found in separate directories for each level of the game (the Shared directory for all shared textures).
Furthermore all TileMap information about each level is saved as .csv-files. The csv-file also contains information about where to find the textures for the TileMap as well as position information for other objects in the level (such as enemies or doors). Each time you enter a new level the csv-files are loaded by that level.

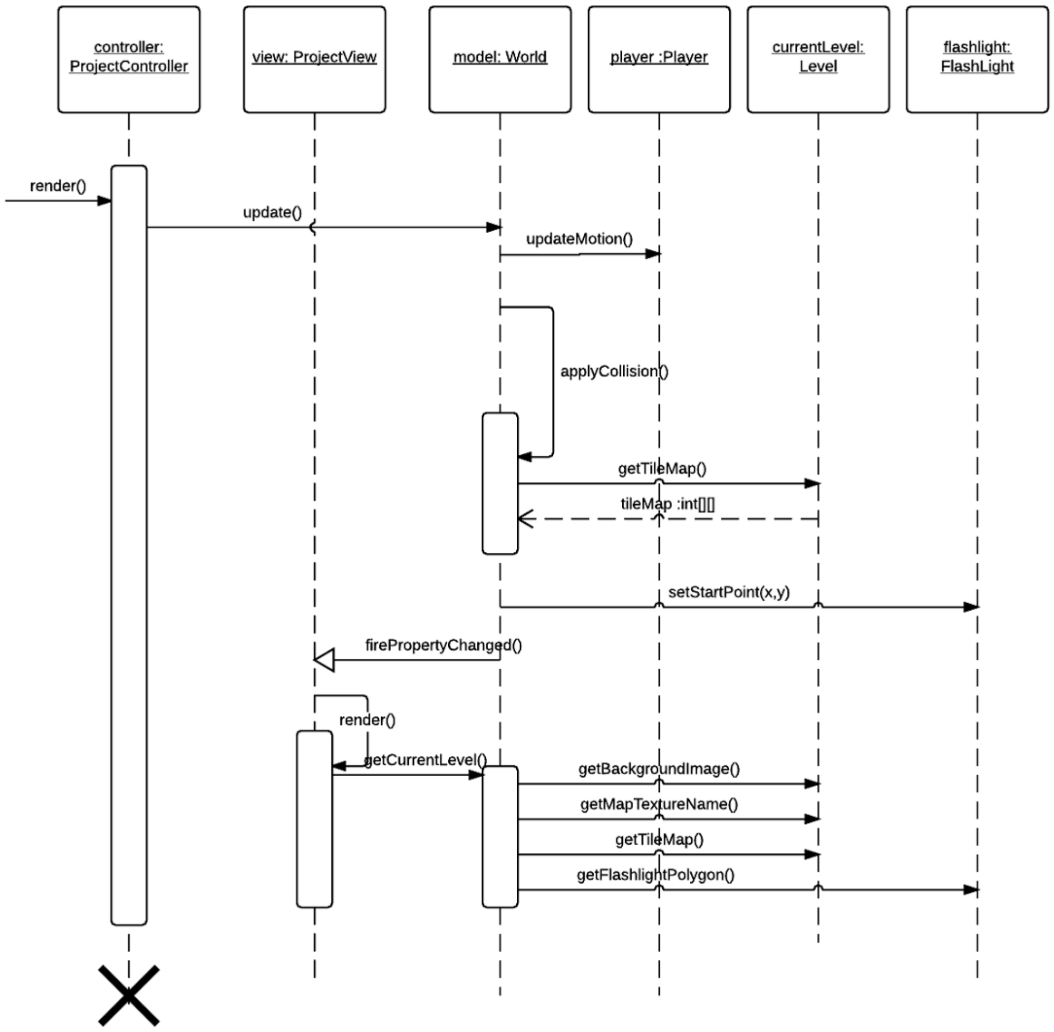
2.5 Access control and security
N/A

2.6 Boundary conditions
N/A

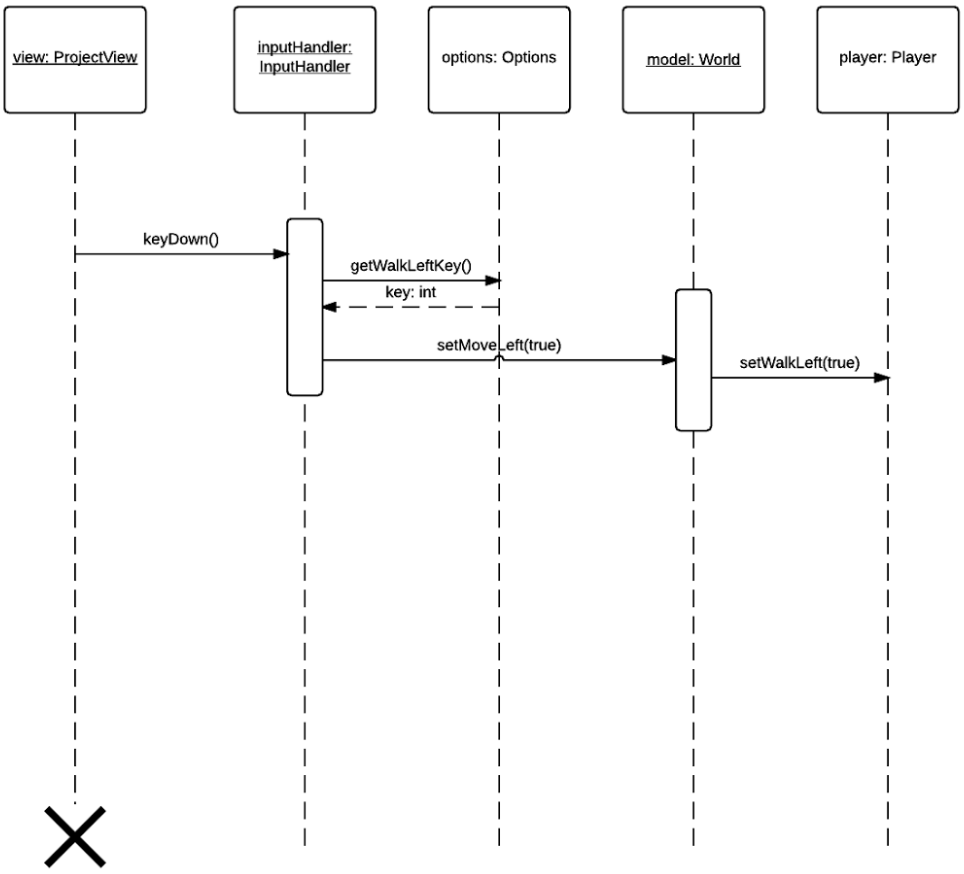
3 References
N/A

APPENDIX

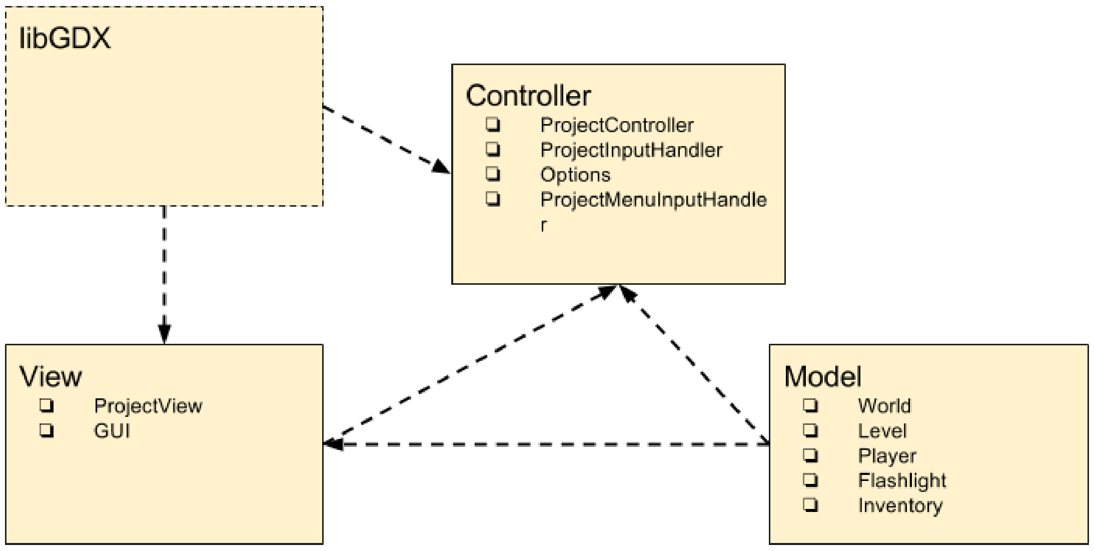
A.1: Sequence diagrams:
Main loop:



Input sequence: (key to walk left pressed)


view: ProjectView
inputHandler: InputHandler
options: Options
model: World
player: Player
keyDown()
getWalkLeftKey()
key: int
setMoveLeft(true)
setWalkLeft(true)

A.2 Package diagrams


libGDX
Controller
ProjectController
ProjectInputHandler
Options
ProjectMenuInputHandle
r
View
ProjectView
GUI
Model
World
Level
Player
Flashlight
Inventory