System design document for Project Dimensions (SDD)

Contents

Version: 1.0

Date: 2013-05-19

Authors: Kim Egenvall, Carl Fredriksson Simon Bengtsson, Kim Kling

1 Introduction

1.1 Design goals

The game will be designed with as loosely coupled parts as possible to make it easy to add new functionalities and change old. The game is designed to be testable with the opportunity to isolate parts. See RAD for information on usability.

1.2 Definitions, acronyms and abbreviations

Technical vocabulary:

- GUI - Graphical User Interface
- Java - platform independent programming language
- JRE - the Java Run time Environment.
- MVC - a design pattern where all classes belongs to one of the three categories model, view or control. Model classes contain all the data and relevant methods that describes the different states of the application, whereas the view classes contains all the GUI components. The controller classes handles the applications actions - controls the program - by handling the communication between the different classes. All definitions regarding poker follow standard poker vocabulary

- Jump - A action by the player, to navigate between platforms by jumping between them.
- Dimension-switch - When the two dimensional world
- Player - Our main character who is a three dimensional figure, stuck in a two dimensional world.
- Camera - The view on the screen. Can be rotated and moved to see other parts of the world and dimensions.
- Powerup - An item that, for a limited or unlimited time, changes a property of the player or the world.
- Platform - An obstacle that can be jumped on. Used to jump to and from to transport one self from a location to another.
- Dimension - A viewport location. E.g, a dimension could be when you look from the side on the X-Y axes. Another can be from upside on the X-Z axes.

2 System design

2.1 Overview

The application will follow a type of the MVC pattern. The most important thing is that the model is completely separated and can be easily taken out and be used with another framework for example.

The application uses the framework LibGDX. It is built up with different screens and each screen has the methods create, render, dispose and few others. Create is basically like a constructor, setting up the initial state of the application. Render is a method that is running every frame update on the active screen if not told otherwise. Dispose takes care of the garbage collection of assets no longer used.

2.1.2 Law of Demeter

We tried implementing our classes to follow the "Law of Demeter". What it says is basically that classes should know as little as possible about other classes. In the MVC pattern it is recommended that the controller knows everything about the model however which also is the case in our application.

2.1.3 Event handling

In this application event handling is processed by the game loop. When something happens the state of an object changes which is then recognized and handled by interested classes the next loop.

2.2 Software decomposition

2.2.1 General

The system is decomposed into the following modules:
Controller package:
Basically everything that has to do with the controller of the MVC pattern
- Dimensions: The Game which creates all screens

Screen package:
- AbstractMenuScreen : General class of the MainMenu-,Credits-,Customize-,GameOver-,Game-,LevelSelect-, and WinScreen.
- GameScreen: The main controller for the MVC
- CreditsScreen: The screen for displaying our names
- CustomizeScreen: For further implementation, supporting new textures to a player
- GameOverScreen:
- MainMenuScreen
- WinScreen

Model package:
The actual items in the game
- GameModel: The model for MVC
- GameObject: The type of objects in the game such as platforms, powerups and the player
- Enum Dimension: XY,XZ,YZ
- OptionsModel: Options for the PC version that applies settings
- GameObject: Abstract class describing an ingame object
- GameWorld: The actual world, responsible for updating states
- Vector 3: Describes object in the 3-dimensional space
- Level: Holds a list of GameObjects and a TiledMap string, together creating a playable level
- LevelHandler: Holds all created levels, responsible for handling them out. Singleton
- MapHandler: Interface with methods to check if a tile is an obstacle or ground
- PowerUpHandler: Responsible for using the correct powerup that is picked up
- Platform: A platform in 3-dimensional space
- Player: The actual player
- PowerUpHandler: Interface forcing powerups to implement use method
-WorldListener: Interface to update a worldChanged event
- SoundObserver: Interface forcing implementers to have a playsound method
- CollisionHandler: Deals with collisiondetection of GameObjects
-TileCollisionHandler: Deals with collisiondetection between played and the tiles of the TiledMap
- Chaser: Computer player who is chasing the users player
- CheckPoint: A checkpoint on a map that can be returned to

Powerup package:
Contains different kinds of powerups.
- Interface PowerUp
- LowGravity
- Speed
- Dimension Change

Util package:
Contains Constants for name and version.
Assets for handling file path association.
Storage for saving and loading the game progress.

TiledMapHandler for checking specific cells in the TiledMaps.

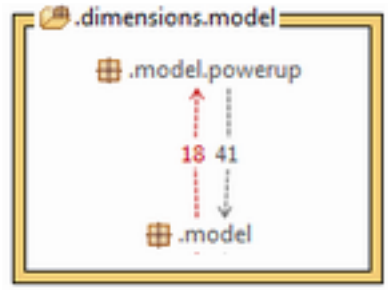
View package:
Contains classes for drawing.
- GameView: The main View for the MVC pattern.
- GameLayerView: Responsible for drawing a separate layer containing text and progress.

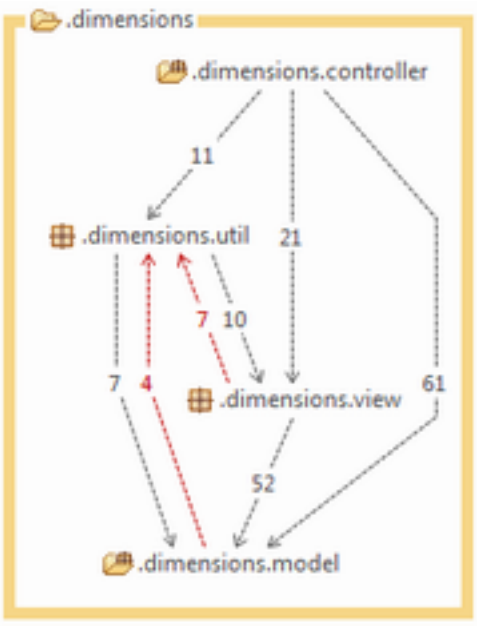
2.2.2 Decomposition into subsystems

TiledMap handling, see Tiled Map Editor under references.

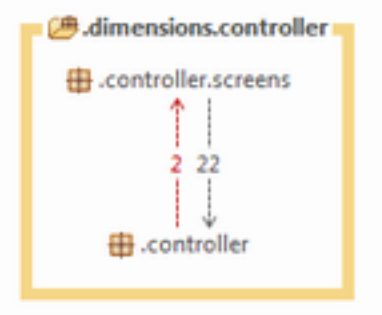
2.2.3 Dependency analysis

Dependencies are shown in figure 1, 2, 3 and 4. They are generated with the tool STAN.





Left: Figure 1 - Model package depenencys
Right Figure 2 - Dependencies in core package



Left: Figure 3 - Dependencies in the controller package

2.4 Concurrency Issues

This is a single threaded application. The thread will be managed by Libgdx and there should be no concurrency issues.

2.5 Persistent data management

For this prototype nothing is saved to file. This could be updated in a later version.

References

LibGDX - http://libgdx.badlogicgames.com/
Tiled Map Editor - http://www.mapeditor.org/
Stan - http://stan4j.com/

Appendix

4.1 Design Model

4.2 Controller Communication Model