SYSTEM DESIGN DOCUMENT FOR SPACE ENTREPRENEURS

CHALMERS TEKNISKA HÖGSKOLA
TDA367

DATE 2017-05-28

AUTHOR: JOSEFINE SVEGBORN, NIKLAS OHLSSON, NIMA AHMADYAN,
SATHIAN SUGUMARAN

1 Introduction

The purpose of this project is to create a 2D arcade game. It will be a single player game with an AI opponent. The game will be similar to the popular arcade game “Space Invaders”, which means that it will include a character shooting down a set of monsters that moves towards the character. To differentiate our game from the existing ones we will have a different interface, with different characters, levels, etc.

The purpose of the game is simply to entertain. The idea is not that it should be played for long sessions and that you should be able to pause and save, but rather it should be simple and game rounds usually not exceeding 5 minutes. Once you lose, you have to start a new game round and start over from the beginning. The game should also include a high score list to give a competitive incentive.

1.1 Design goals

This document describes the construction of the SpaceEntrepreneurs application specified in the requirements and analysis document.

Some design goals:

- The design must be loosely coupled to make it possible to switch GUI and/or partition the application into a client-server architecture.
- The design must be testable i.e. it should be possible to isolate parts (modules, classes) for test.

See RAD for usability etc.

1.2 Definitions, acronyms and abbreviations

Definitions regarding the core and the different animations in the game of “Space Entrepreneurs” are defined below:

- **Monster,** The enemies. They move closer towards the game character and drop missiles. The character is supposed to shoot them down in order to move on to the next level.
- **Player,** The actual player behind the screen.
- **Character,** Is the character, a spaceship, that the player controls through the keyboard.
- **Level,** The different degrees of difficulties, the different levels will contain different amount of monsters, moving in a different pace etc.
- **Cover,** Where the character can take cover from missiles.
- **Missile,** The “projectile” that is fired from the weapons and by the Monsters..
- **Healthpoints,** The number of chances the player has, number of times the character can get hit before the game is over.

See RAD document for more definitions etc.

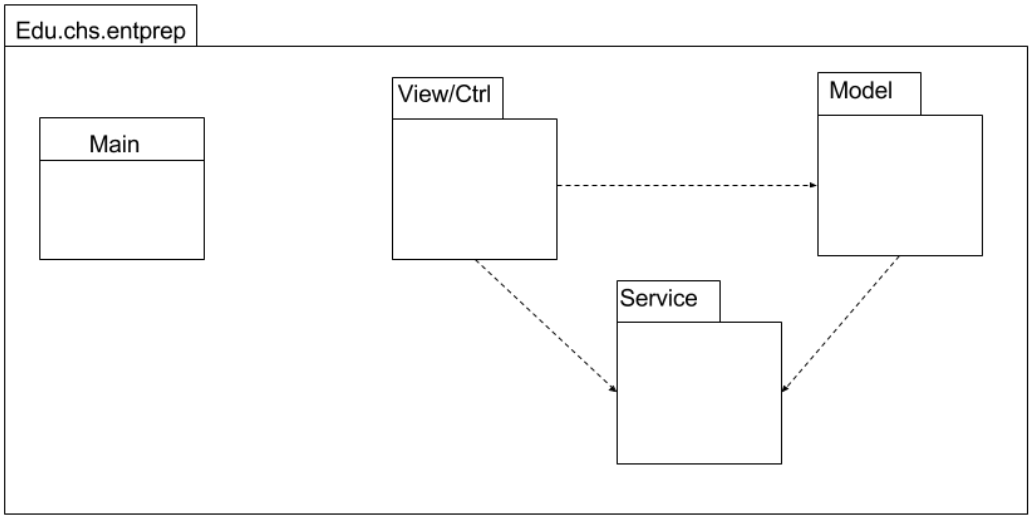
2 System architecture

This chapter will be an overall description of the system.

2.1 Overview

The application will be based on an MVC inspired model used especially for game development and the application run on a single desktop computer.

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



- Main: is the application entry class, which initiate the game view/controller.
- Model: contains the OO-model and is the core of the project.
- View/Ctrl: controls the events and contains all GUI related classes
- Services: handles images and sounds.

2.2 General Observations

Functionality

The main class initiates the MenuView, where the start button allows the player to build the different game objects needed. The GameView class can be seen as a root class where all other applications in the model are called upon.

KeyEvents - Input list

To handle the different events from the keyboard the different key events is saved in an input list. This is done in order to create the best user experience for the player and being able to better control the character's movement (i.e. spaceship in game environment). For example when pressing left and right keyboard buttons the spaceship stands stills and waits until one or both of the buttons is released.

Sprite

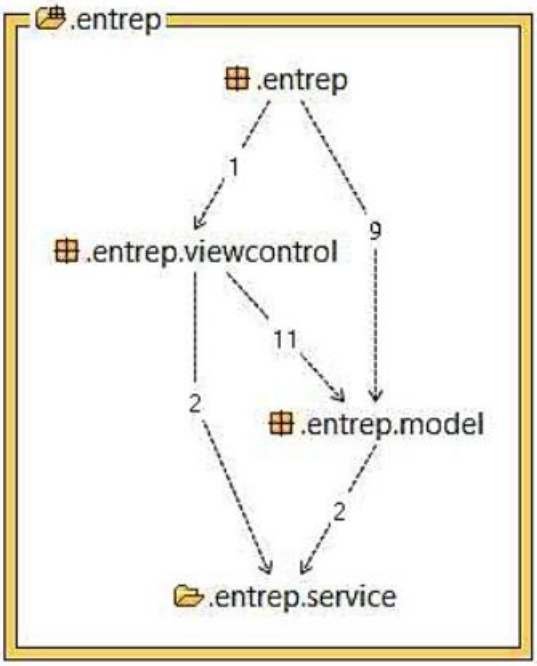
Most of the objects in the application are so called Sprite-objects (an instance of Sprite). The Sprite class holds all attributed of the different objects. It also contains all methods for handling the velocity, position, etc. for the different objects.

Animation Timer

The game is handled by a looping timer that renders and updates the logic. This is the heart of the application which both handles communication between the KeyEvents input list from the gameView to initiate the logic in the model.

Dependency analysis

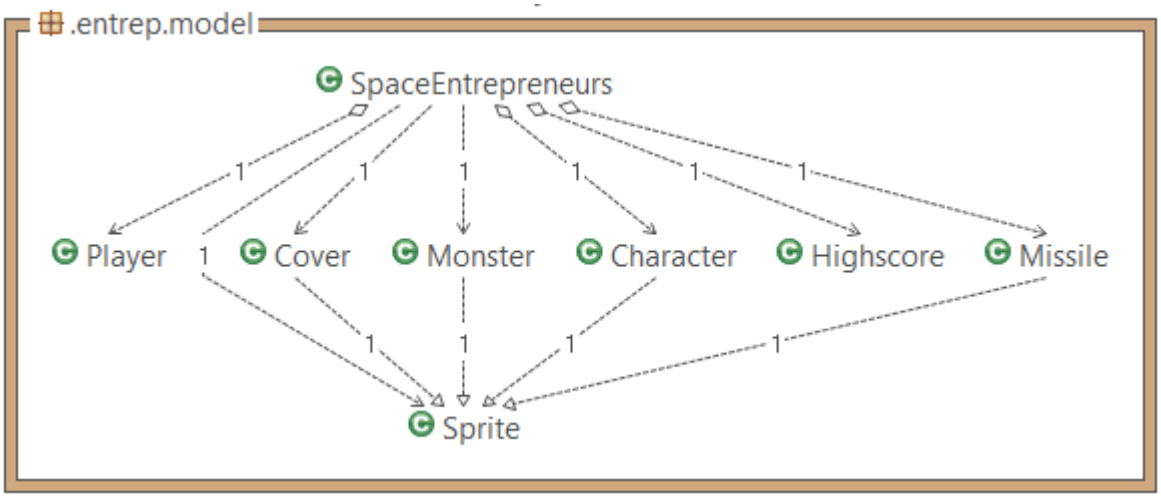
Dependencies are illustrated in the figure below.



3 Subsystem decomposition

The system consist of two different services one which is handling the images responsible for getting images, background images, symbols etc. and the other handling the audio responsible for the game background sound and different sounds for different events.

3.1 Model (Core)



The dependency analysis above visualizes the model of the game. Where it is seen that the different objects built are dependent on the Sprite class. The SpaceEntrepreneurs contains the constructor building the game. The SpaceEntreprenuers constructor is called from the GameView class which decides its parameters, player, level etc.

3.2 Audio and Image service

The service packages that have been developed contains of an audio handling and an image handling. The audio service let us use different sounds and background sound when different events happen in the game, e.g. when the character shoots a missile a specific sound file is played. The two different services consist of an Interface, factory that decide what the service needs to include. Booth the different services exist of a map structure, which makes it easier to being able to switch the images or audio, being able to download the images that we want to use directly from a webserver or database. This is also a benefit if the program will be converted into another platform e.g. mobile phone then the different images sizes etc. need to be changed, which the service conduct.

3.3 Test files

The application comprises of three test classes that test the different intersect methods, highscore update and the transition to next level. The test classes can be found at edu.chs.entrep.test.

4 Persistent data management

The data management is needed to be handle as the next step in the project. As for now in the development phase all data such as the highscore.txt file, sounds, images, icons etc. are stored locally.

5 Access control and security

No special control or security depending on the player. The application is launched and exist on the screen as a normal desktop application (using scripts).

6 References

1. SpaceInvadors Game: https://sv.wikipedia.org/wiki/Space_Invader
2. MVC, see http://en.wikipedia.org/wiki/Model-view-controller