Requirements and Analysis Document for SpaceEntrepreneurs

Version: 1

Date: 2017-05-28

Author: Nima Ahmadyan, Niklas Ohlsson, Sathian Sugumaran, Josefine Svegborn.

This version overrides all previous versions.

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 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 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 loose, you have to start a new game round and start over from from the beginning. The game should also include a high score list to give a competitive incentive.

1.1 Definitions, acronyms and abbreviations

Technical definitions

- GUI, graphical user interface.
- Java, platform independent programming language.
- JavaFX, set of graphics and media packages.
- JRE, the Java Runtime Environment. Additional Software needed to run an Java application.

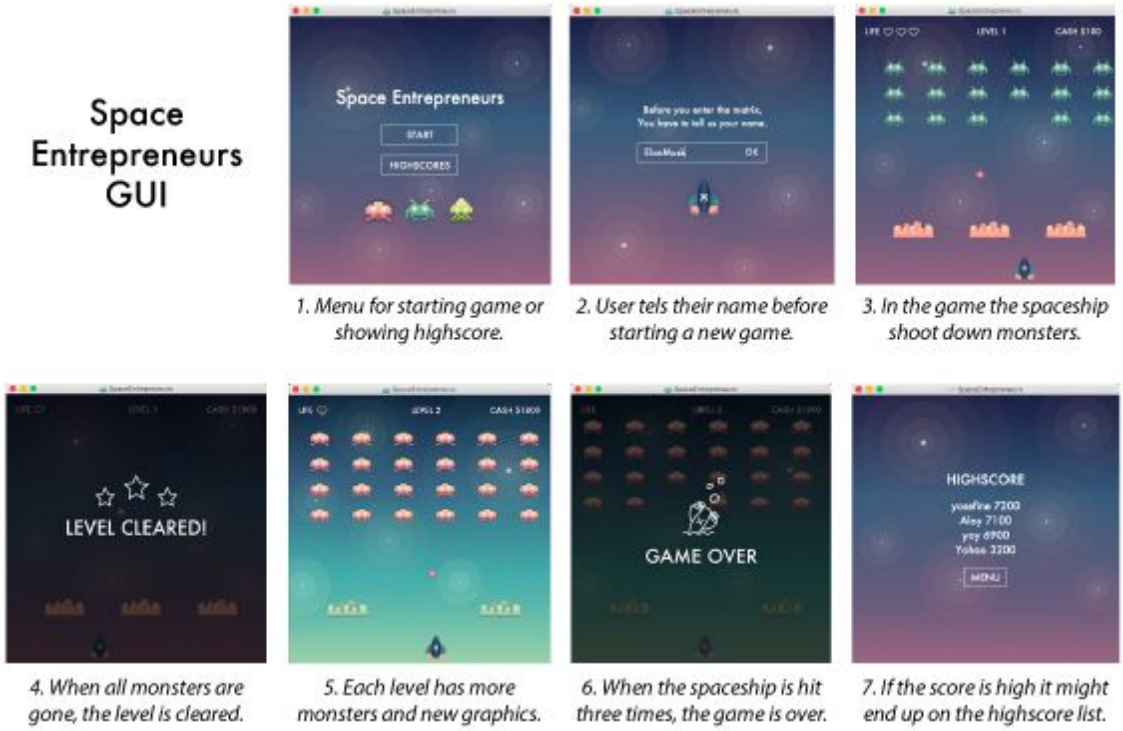
Definitions regarding the core and the different animations in the game of SpaceEntrepreneurs are defined below:

- **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 though the keyboard.
- **Level,** The different degrees of difficulties, the different levels will contain different amount of monsters, different graphics 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, number of times the character can get hit, has before the game is over.

2 Requirements

2.1 User interface - Graphic user interface



1. Menu for starting game or showing highscore.
2. User tels their name before starting a new game.
3. In the game the spaceship shoot down monsters.
4. When all monsters are gone, the level is cleared.
5. Each level has more monsters and new graphics.
6. When the spaceship is hit three times, the game is over.
7. If the score is high it might end up on the highscore list.

Menu, Name & Highscore
Buttons are indicated with a whiteline. The buttons glow when hovering and there is a sound feedback when clicking. When writing the name, it is also possible to use the Enter-key instead of pressing the OK-button.

Game
In the top panel of the window the game status can bee seen. Health is indicated to the left with 1-3 hearts, level is in the middle and score is called cash in the right corner. If the player moves on to the next level or gets game over, this is indicated with a layover image showing for 3 seconds before the game moves on. Each new level has new graphics clarify that the game moves forward.

2.2 Functional requirements

The following use cases should all be possible and are further elaborated in chapter 3.

Player should be able to:

1. Start Game
    a. Type in Name
    b. Move and Shoot using the keyboard (left, right and space)
2. See Highscore
3. Close Application at any given time

Ordering use cases by priority

- Move Character
- Character shoots
- Hit Monsters
- Monster Move
- Monsters reached lower boundary
- Game over
- Monster shoots
- Character damage
- Start game
- Exit
- Player name
- High Score
- Next Level

2.3 Non-functional requirements

2.3.1 Usability

The game should not require any advanced previous knowledge and should be intuitive for the player. However, it is expected that the player will know standard control of a 2D-game - moving with arrows and shooting with space.

2.3.2 Performance

The game should go about without lagging and should react with the players decisions to move and/or shoot

2.3.4 Supportability

The application will be compatible with both Windows and Mac.

2.3.5 Technical

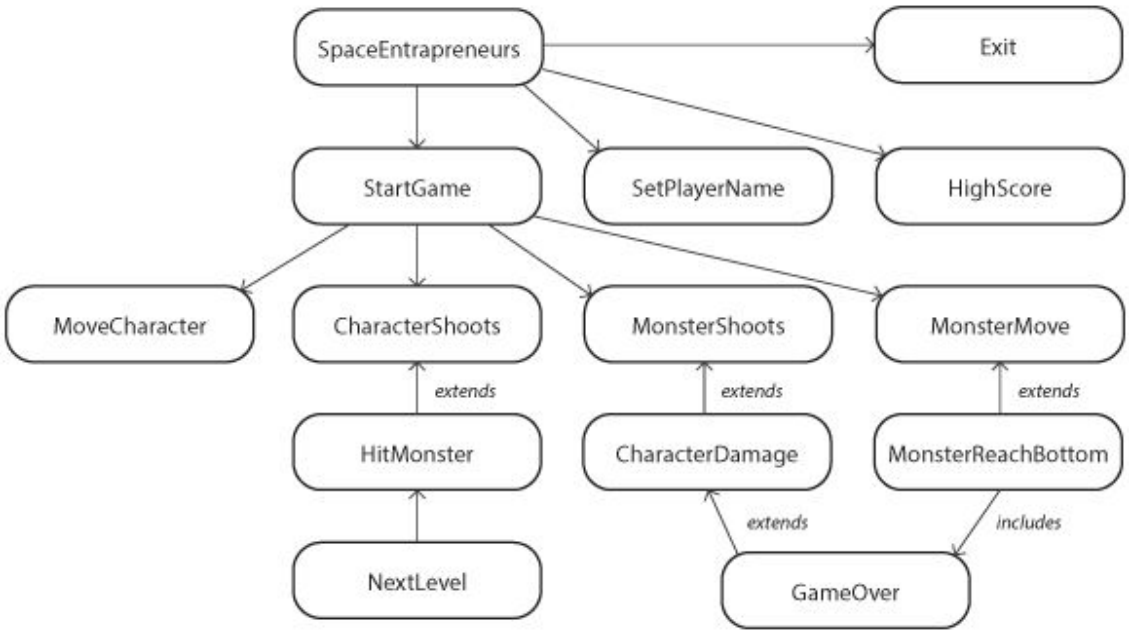
There should be automated test for the important use cases, and all code could be tested through the user interface.

2.3.6 Legal

The images used for the game is created by the group and they have therefore full copyright. The soundfiles are borrowed, but since the game is solely for educational purposes, this should be a problem.

3 Use cases

The following Use Case shows how the use cases are related to each other.



3.1 Use case listing

Use case texts (using the use case template)

1. Move Character
Summary: Game has started, and the player moves the character.
Priority: High
Extends:
Includes:
Participators: Actual player

	Actor	System
1	Player press right or left keyboard button	
2		Character moves to right or left depending on which button have been pressed
2.1	If, character reach right or left border (edge of the screen)	
2.2		The system blocks/stops the character to move further in that direction

2. Character Shoots

Summary: The player fires a missile through the character during the game.
Priority: High
Extends:
Includes:
Participators: Actual player

	Actor	System
1	Player press space	
2		A missile moves straight up, starting from the character's position. Indicate this through a shoot sound.
2.1.1 (Out of screen)		
2.1.2		Missile removed from screen
2.2.1(Hits Obstacle)		
2.2.2		Missile disappears (fixed obstacle)
2.3.1(Hits Monster)		
2.3.2		See **Hit Monster**
2.4.1(Holds space)	Holds space	
		Check if any missile is on screen. if not, see 2(character shoots).

3. Start game
Summary: The first screen when starting the application
Priority: mid
Extends:
Includes: Move Character, Character Shoot, Monster Shoot, Monster Move
Participants: Actual player

	Actor	System
1.		Display startpage
2	Presses Start	
3		Screen changes and ask player for name. Which the player can type in a textbox.
4.	Types in name	
5.	Press ok	
5.		System checks if the input is a string and contains at least one keyboard input
6. Ok input		Game interface starts and background music
6.2 No input		A message to the player comes up, and describe that you need to input your name, see 4 (start game)

4. Game over

Summary: The screen that shows when the character loses.
Priority: mid
Extends: Character Damage
Includes: -
Participants: Actual player

	Actor	System
1.		Screen Freezes
2.		Prints “Game Over”
3 (Don’t reach high score top 4)		
3		New “HIGH SCORE”

5. Character damaged
Summary: Character gets hit from invaders during the game
Priority: mid
Extends: Monster Shoots
Includes:
Participators:

	Actor	System
1. (Character gets hit from invaders)		Remove 1 heart from existing amount of hearts and indicate this through flashes and sound
2(If the last heart)		See **Game over**

6. Monsters reached lower boundary
Summary: The case when the the monster reaches the lower boundary, when the player does not manage to shoot down the monster before they reach the character.
Priority: mid
Extends: Monster Move
Includes: Game Over
Participators:

	Actors	System
1. Monster reaches the lower boundary		See Game over

7. Monster move
Summary: The monster moves downward during the game
Priority: High
Extends:
Includes:
Participators:

	Actors	System
1.		Monsters moves downwards and sideways
1.1 (Invader reaches the boundary)		See **Monster reached lower boundary**

8. Hit Monster
Summary: Missile intersect with Monster.
Priority: high
Extends: Character Shoots
Includes:
Participators:

	Actor	**System**
1.		Remove monster and a “boom” sound indicates the hit.

9. High Score
Summary: Missile intersect with Monster.
Priority: low
Extends:

Includes:
Participators:

	Actor	**System**
1.1 (if top 4 score)		Update high score list with player name and score and show congratulation image.
1.2 (if not top 4 score)		Show high score list without congratulation image.
3.	Press “MENU”	
4.		Displays startpage with menu

10. Next Level
Summary: When all the monsters are destroyed, the game move on to next level.
Priority: low
Extends: Hit Monster
Includes:
Participators:

	Actor	**System**
1. If all monsters are dead		Initiate next level background and music
2.		Set new level background with more monsters and new covers

11. Player Name
Summary: When all the monsters are destroyed, the game move on to next level.
Priority: low
Extends:
Includes:
Participators:

	Actor	**System**
1.	Type in player name	
2.if player name doesn’t contain input		print out “type again“
3. if player name valid		see Start game

12. Monster Shoot
Summary: When all the monsters are destroyed, the game move on to next level.
Priority: low
Extends:
Includes:
Participators:

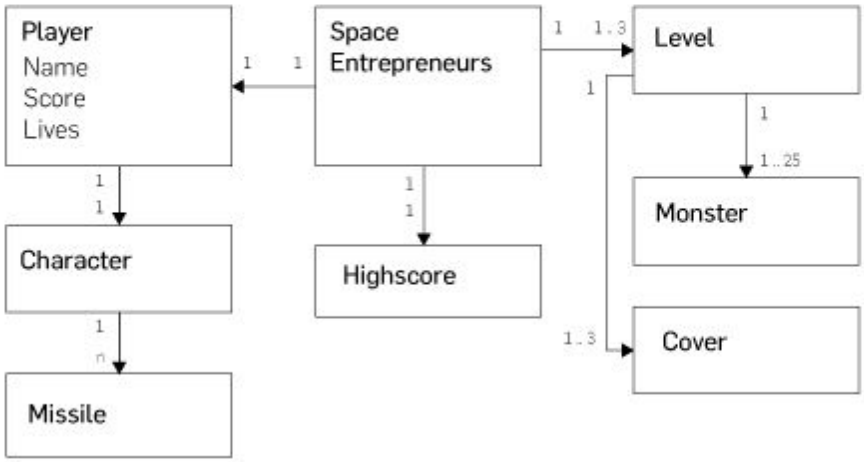
	Actor	**System**
1.		fires missiles from random monster at random time
2.		

13. Exit
Summary: When all the monsters are destroyed, the game move on to next level.
Priority: low
Extends:
Includes:
Participators:

	Actor	**System**
1.	if pressed exit button	
2.		Shut down game and java file

4 Domain model

From the domain model below it can be seen how the SpaceEntrepreneurs class controls the game round. The player, character and missile remains the same throughout a game round, while the monsters and covers change depending on the level. The high score is separate from the other game functions.



4.1 Class responsibilities

Main: Is the overall representation of the game, initiates the menu page.

SpaceEntrepreneurs: Is the initiator of the game, this starts when the SpaceEntrepreneurs have started a new game. SpaceEntrepreneurs will be in control of the logic of the game e.i. when to move into a new level.

Player: Is a representation of the human player, holds the Player's name, Score and Life.

Character: This is the animated spaceship that represent the player in the game. The character should be able to move, have a weapon and shot missiles which is solved by the game logic in SpaceEntrepreneurs.

Missile: Will control and hold the position of the missile(s) in the space. (Then a certain class collision check/control, will take care of the collisions of missiles, covers, and monsters.)

Level: Represent the different available levels in the game, three in total. Be in charge of what's included in the different levels, such as amount of monsters, covers and bosses.

Monster: This class will represent the monsters, and inherits the methods from sprite to locate their movement in the space, the monsters lives, colors etc.

Cover: This represent different covers in the space, where the character can hide from the monsters missiles.

Sprite: This class is a parent class for all interactive game objects (character, monster, missile, cover, etc. It contains methods and attributes for managing position, velocity, checking intersection, etc.

Highscore: this class manage the highscore list and logic.

GameView: This class represents the view and control for the actual game. It initiates the player and a new SpaceEntrepreneurs class for each level. It includes event handlers for controlling the character through key presses. It also contains an animation timer in which all graphics are updated and game status is checked.

HighscoreView: This class represents the view and control for the highscore scene. It reads from a highscore file and prints the list to a label. Launches the menu scene if the menu button is pressed.

MenuView: This class represents the view and control for the menu. It launches name view or highscore view if those buttons are pressed.

NameView: This class represents the view and control for the scene in which the player inputs its name. It launches the GameView class.

5 References

- Java 8
  http://docs.oracle.com/javase/8/docs/
- JavaFX
  http://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm#JFXST784