ALL REQUIREMENTS OF SPACE INVADERS

FINAL VERSION

by

A. Smesseim, S. Baraç, A. el Khalki, A.W.L. Oostmeyer, M. Maton

in partial fulfillment of the requirements for the degree of

Bachelor of Science
in Computer Science

at the Delft University of Technology,

to be presented on 30 October 2015.

Supervisor: Dr. A. Bacchelli

CONTENTS

1 20-Time, Reloaded **1**

1.1 Requirements 1

1.1.1 Functional Requirements 1

1.1.2 Non-functional Requirements 4

Bibliography **5**

1

20-Time, Reloaded

1.1. Requirements

1.1.1. Functional Requirements

For the game Space Invaders, the requirements regarding functionality and service are grouped under the Functional Requirements. Within these functional requirements, four categories can be identified using the MoSCoW[1] model for prioritizing requirements.

Must Haves

1. The game must show all 60 enemies at the start of the game.
2. The enemies are aligned in rectangle of 5 rows and 12 columns.
3. The rectangle of enemies starts at the top left of the playing field.
4. The player and the enemies must not be able to move outside the boundaries of the playing field.
5. The player must be able to move left and right using the arrow keys.
6. The player must be able to shoot bullets up.
7. If the player's bullet hits an enemy, the enemy disappear from the playing field.
8. The score should be shown above the playing field.
9. There should be three different kinds of enemies, with different appearances. The bottom two rows consist of enemies named "large enemies". The top row should consist of enemies named "small enemies". All other rows should consider consist of enemies named "medium enemies".
10. The player could be able to shoot a bomb up using keyboard input, if the player has any bombs. If a bomb is fired, the number of bombs is reduced by one.
11. If a player bullet hits an enemy bullet, then both bullets could disappear.
12. A UFO could appear at either the top left or the top right of the playing field. The starting point of the UFO is randomly determined, and the time of appearance is also randomly determined. The UFO can only move horizontally. When the UFO appears, will move to the other side of the playing field at the speed of one length unit per second, where the length unit is half the width of the UFO. If a UFOis hit by a player bullet, then a random number of points between 100 and 1000 is awarded to the player. When the UFO reaches the other side of the playing field, it disappears.
13. The logger must output the actions of every game entity. Each individual description of an action is called a log.

14. The format of a log for moving actions is: `<timestamp> - INFO: <type> moved from (<oldX>, <oldY>) to (<newX>, <newY>)`, where `<timestamp>` is the time the action was performed (conforming ISO 8601), `<type>` is the type of entity that moved. `<oldX>`, `<oldY>`, `<newX>`, `<newY>` are the old x-coordinate, the old y-coordinate, the current x-coordinate and the current y-coordinate, respectively.

15. The format of a log when a bullet is fired is: `<timestamp> - INFO: <type> fired a <bullettype> at (<X>, <Y>) in the direction <direction>`, where `<timestamp>` is the time the action was performed (conforming ISO 8601), `<type>` is the type of entity that fired the bullet, `<bullettype>` is the type of bullet that is fired (either a bullet or a bomb). `<X>`, `<Y>` and `<direction>` are the x-coordinate and y-coordinate the bullet was fired from, and the direction the bullet is fired to, respectively.

16. The format of a log when a bullet has hit an entity is: `<timestamp> - INFO: <type> is hit by a <bullettype> at (<X>, <Y>)`, where `<timestamp>` is the time the action was performed (conforming ISO 8601), `<type>` is the type of entity that was bit by the bullet, `<bullettype>` is the type of bullet (either a bullet or a bomb). `<X>`, `<Y>` and `<direction>` are the x-coordinate and y-coordinate the bullet was fired from, and the direction the bullet is fired to, respectively.

17. The preferred output stream (where the description of the actions will be written to) must be specifiable by the caller. In the implementation of the logging feature in the game, the logger will print all logs/descriptions to stdout.

18. Four barricades must be placed between the enemies and the player, equally spaces horizontally over the playing field.

19. The barricades must not let bullets pass through them.

20. The barricades must be colored green.

21. 'Ave Maria' must be played as background music.

22. The size of each barricade must be 200x100 pixels.

23. Four barricades must be placed between the enemies and the player, equally spaces horizontally over the playing field.

24. The barricades must not let bullets pass through them.

25. The barricades must be colored green.

26. 'Ave Maria' must be played as background music.

27. The size of each barricade must be 200x100 pixels.

28. The game must show the current level above the playing field. At the start of the game, the current level is 1.

29. If all enemies have disappeared from the playing field (i.e. are killed), then a store is shown. This store is text-based, and allows the player to trade some of its points to purchase powerups. The powerups are:

    - an additional life for 1000 points.
    - a bomb, that lets the enemy disappear upon impact, as well as all the horizontally, vertically and diagonally adjacent enemies, for 250 points.
    - the restoration of the barricades to their original state, for 500 points.

30. The store must show a "Continue" button, that terminates the store, and shows the next level to the player. The enemies and ship are reset to their starting position, and the level counter is incremented.

31. The game must show a "Save game." button when the user is in the store.

32. When the player presses this button, the number of lives, bombs, points, the current level and the intact (non-destroyed) barricades are written to the file named "spaceinvaders.sgfile". The text of the button changes to "Save game. Done!".

33. The game must show a "LOAD GAME" button in the main menu.

34. If the player presses on this "LOAD GAME" button, and the save file "spaceinvaders.sgfile" is available, then game shows the store, with the exact same values for the number of points, bombs, lives and the current level as the last time the player pressed "Save game.". If the player then presses "Continue", then the game shows the same barricades as at the end of the level before "Save game." was pressed.

SHOULD HAVES

35. A log level of a log/description should be specifiable by the caller. The log level is the priority of the particular log/description. At least three levels should be specifiably, namely Info, Warning, and Error. In the implementation of the logging feature in the game, all logs of actions will have the log level Info.

36. The minimum log level that warrants writing the log/description to the output stream should be specifiable. In the implementation of the logging feature in the game, the logs that have the log level Info, Warning or Error will be written to the output stream.

37. If a bullet hits the barricade, then only the point of intersection of the barricade should be damaged (i.e. destroyed).

38. If a bullet hits the barricade, then only the point of intersection of the barricade should be damaged (i.e. destroyed).

39. The speed of the enemies should be cx where x is the current level, and c is a preset constant.

40. If the player presses on this "LOAD GAME" button, and the save file "spaceinvaders.sgfile" is available, and then presses "Continue", then the game shows the same barricades as at the end of the level before "Save game." was pressed.

COULD HAVES

41. A sound effect could be played when a player fires a bullet or a bomb.

42. In addition to lives, bombs, and the restoration of the barricades, the store could also offer:

    - the ability for the player to move vertically, for 2000 points.
    - a shield, where the player is invulnerable at the start of the next level for 20 seconds, for 300 points.

43. Instead of having a text-based store, the store could also show sprites near the offered items. To clarify:

    - instead of showing "Buy a life", the store could show a heart sprite.
    - instead of showing "Buy a bomb", the store could show a bomb sprite.
    - instead of showing "Buy a shield", the store could show a shield sprite.

44. If the player presses on this "LOAD GAME" button, and the save file "spaceinvaders.sgfile" is not available, then the text "LOAD GAME" changes to "LOAD GAME (failed).

WOULD/WON'T HAVES

45. The game won't support saving the player's progress after each level.

46. The game won't support saving the player's progress to a different file than "spaceinvaders.sgfile".

1.1.2. NON-FUNCTIONAL REQUIREMENTS

In addition to the functional requirements specified in chapter 1.1.1, the game should also adhere to several non-functional requirements, outlined in this chapter. These requirements will not affect the functionality of the game, but will ensure that the game is gradable by the instructors of the course TI2206. The non-functional requirements of the game are:

47. The game must be playable on Windows (7 or higher), Mac OS X (10.8 and higher) and Linux.
48. The game must be implemented in Java.
49. A first fully working version of the game must be delivered on September 11, 2015
50. Scrum methodology must be applied.
51. The source code must be hosted on GitHub as a public repository.
52. The version control used for developing this game must be Git.
53. The tests must be automated using JUnit.
54. This project must use Maven for build automation.
55. This project must use Travis CI for continuous integration.
56. This project must employ the following static analysis tools: Checkstyle, PMD and FindBugs.

Bibliography

[1] S. Ash, *Moscow prioritisation briefing paper,* DSDM Consortium (2007).