Space Invaders : Requirements

Group 22
TI2206 Software Engineering Methods

Ege de Bruin
Bryan van Wijk
Martijn van Meerten
Dorian de Koning
Jochem Lugtenburg

October 13, 2015

Supervisor: Dr. A. Bacchelli
TA: Danny Plenge

Delft University of Technology
Faculty of EEMCS

Contents

1 Functional Requirements

A list of functional requirements which were considered during the planning of the project. Functional requirements describe services or functions a system should deliver to the end user. The requirements are listed using the MoSCoW method. The MoSCoW method is a widely used standard in the industry and involves prioritized requirements. The requirements are categorized in four categories, 'Must', 'Should', 'Could' and 'Would/Won't'. If a 'Must' is not included in the final deliverable, the project is not considered a success.

1.1 Must Haves

The Space Invader game must meet the following requirements:

- The game shall start when the player presses a predefined key on the keyboard.
- The game shall load a grid of 10 x 4 enemies in the form of aliens before a new game starts.
- The game shall load a player in the form of a spaceship on the bottom of the screen before a new game starts.
- The player shall have three lives after a new game has started.
- The player shall lose one life when it is hit by an enemy bullet.
- The enemies shall move horizontal in between the borders of the game screen, starting in the middle of the screen, from left to right.
- The enemies shall start with a movement speed of 40 pixels per second.
- The enemies movement direction shall be reversed upon hitting a border of the game screen.
- The enemies shall move 10 pixels downwards upon hitting a border of the game screen.
- The game shall end if the player's lives are depleted.
- The game shall end if the enemies reach the same height as the player.
- The player shall be able to move his/her character on a horizontal axis to the left and right using predefined keys.
- The player shall be able to shoot bullets on a vertical axis towards the top of the screen using a predefined key.
- The enemies shall shoot bullets on a vertical axis towards the bottom of the screen.
- An enemy shall disappear if it is hit by a bullet shot by the player.
- The game shall show the amount of lives on the top right of the screen.

1.2 Should Haves

The Space Invader game should meet the following requirements:

- The player shall be able to restart the game after the game ends.
- The game shall be able to display the overall high score on the screen.
- The enemies shall move 4 pixels per second faster upon hitting the border of the game screen.
- The game shall load 4 barricades between the enemy and the player before a new game starts.
- The barricades shall become damaged after being hit by a bullet.
- The bullet shall disappear after hitting a barricade.
- The player shall be able to pause the game.
- A player shall obtain ten points upon hitting an enemy with a bullet.
- The game shall load with an initial score of zero points.
- The game shall show the total amount of points obtained on the top left of the screen.
- The game shall notify the player on the screen after the game ends.
- The game shall load a new grid of 10 x 4 enemies in the form of aliens if all the aliens are killed.

1.3 Could Haves

The Space Invader game could meet the following requirements:

- The game shall have different levels which can be played by the player.
- The player shall be able to collect power-ups by shooting a red enemy.
- The game shall have a power-up which increases shooting speed for five seconds.
- The game shall have a power-up which increases movement speed for five seconds.
- The game shall play a music track while the game is in progress.
- The game shall have sounds effects if a bullet is fired and if a bullet hits an enemy.
- The game shall display an explosion effect after an enemy or player is hit by a bullet.
- The players movement speed shall increase while moving.
- The player shall bounce back upon hitting the border of the screen.

1.4 Would/Won't Haves

The Space Invader game could meet the following requirements:

- The player shall be able to upgrade its spaceship after a wave of enemies.
- The player shall be able to interact with the game using the mouse.

2 Non-Functional Requirements

A list of non-functional requirements which were considered during the planning of the project. Non-functional requirements are constraints on the system or development process. If the non-functional requirements cannot be met, the system is considered useless.

- The program shall be executable on Windows (7 and higher), Mac OS X (10.8 and higher), and Linux.
- The game shall be implemented using the Java programming language.
- The game shall be implemented using Java version 8.
- The code shall be placed under version control using Git, using a remote repository on GitHub.
- The code shall be compiled using Apache Maven.
- The code shall be analyzed for programming flaws using PMD and FindBugs.
- The code shall be analyzed using Checkstyle before each push to GitHub.
- A first fully working version of the game shall be delivered at September 11, 2015

3 Logger Functional Requirements

A list of functional requirements considered for the extension of the game with a logger using the MoSCoW method described in the previous section.

3.1 Must Haves

The Logger of the game must meet the following requirements:

- The game shall be able to log all actions happening during execution the game.
- The logger shall be able to write the log to a file.

3.2 Should Haves

The Logger of the game should meet the following requirements:

- The logger shall keep track of the time each action is executed.
- The logger shall have different logging levels.
- The logger shall be able to log exceptions thrown.
- The logger shall be able to log errors.

3.3 Could Haves

The Logger of the game could meet the following requirements:

- ...

3.4 Would/Won't Haves

The Logger of the game won't meet the following requirements:

- The logger shall use text colors in the output log.

4 Code Improvement Functional Requirements

A list of functional requirements considered for code improvements using the MoSCoW method described in the previous section.

4.1 Must Haves

The Code Improvements must meet the following requirements:

- Code shall be Checkstyle compliant.
- Code shall be PMD and Findbugs compliant.
- Classes shall only have one responsibility.
- Methods shall not be too long.

4.2 Should Haves

The Code Improvements should meet the following requirements:

- Code shall contain Javadoc comments explaining complex methods.
- Code shall not be commented.
- Static shall be used where appropriate.
- Private access modifiers shall be used where appropriate.
- Test coverage shall be at least 75%.
- Code shall contain less duplicate code.

4.3 Could Haves

The Code Improvements could meet the following requirements:

- Variables shall be named in correct English.
- Methods shall be named in correct English.
- Classes shall be named in correct English.
- HashCode methods shall be implemented correctly where equals is used.

4.4 Would/Won't Haves

The Code Improvements won't meet the following requirements:

- ...

5 Powerup Functional Requirements

A list of functional requirements considered for the implementation of powerups using the MoSCoW method described in the previous section.

5.1 Must Haves

Powerups must meet the following requirements:

- Aliens shall randomly drop a powerup upon death.
- A Powerup buff shall be applied upon collision with the spaceship.
- A Powerup shall disappear upon collision with the spaceship.

5.2 Should Haves

Powerups should meet the following requirements:

- Player velocity shall increase for a feasible amount of time upon collision with a Speed Powerup.
- Player shooting speed shall increase for a feasible amount of time upon collision with a Shooting Powerup.
- Player life shall increase upon collision with a Life Powerup.
- Player life shall not increase if the maximum amount of 5 lives has been reached.
- A Powerup shall have a custom sprite based on its type.
- A Powerup shall not disappear upon collision with a bullet.
- A Powerup shall not disappear upon collision with an alien.

5.3 Could Haves

Powerups could meet the following requirements:

- A Powerup shall explode upon hitting a barricade.

5.4 Would/Won't Haves

Powerups won't meet the following requirements:

- A Powerup shall explode upon colliding with a bullet.

6 Wave Functional Requirements

A list of functional requirements considered for the implementation of different waves using the MoSCoW method described in the previous section.

6.1 Must Haves

The Waves must meet the following requirements:

- A new wave shall have a different pattern of aliens.
- A wave pattern shall be stored in a file.
- Patters shall be loaded upon starting the game.

6.2 Should Haves

The Waves should meet the following requirements:

- Aliens shall have different sizes.
- Aliens shall have different types.
- The game shall start with a standard pattern.
- The game shall load a random pattern upon loading a wave.
- Alien types shall have different colors.
- Alien types shall have different shooting speeds.

6.3 Could Haves

The Waves could meet the following requirements:

- Different alien types shall take more hits before they die.
- Rows of aliens shall move in independent directions of each other.

6.4 Would/Won't Haves

The Waves won't meet the following requirements:

- Aliens shall have different movement speeds.

7 Local co-op Multiplayer Functional Requirements

A list of functional requirements considered for the implementation of local co-op multiplayer using the MoSCoW method described in the previous section.

7.1 Must Haves

The local co-op multiplayer must meet the following requirements:

- A multiplayer game shall contain two spaceships.
- The game shall be able to load a multiplayer game.
- The first spaceship shall be controlled using predefined keys on the keyboard.
- The second spaceship shall be controlled using predefined keys on the keyboard.

7.2 Should Haves

The local co-op multiplayer should meet the following requirements:

- The User Interface shall have a menu to select between single or multiplayer games.
- The User Interface shall display two different score elements and life elements for different players.
- Each spaceship shall have an equal initial movement speed.
- Powerups work for individual spaceships.

7.3 Could Haves

The local co-op multiplayer could meet the following requirements:

- The game shall end after one of both players runs out of lives.
- The game shall display the winner after the game ends.
- The player wins if he/she has the highest score.
- The game shall spawn a powerup with the ability to freeze the other player for a predefined amount of time.

7.4 Would/Won't Haves

The local co-op multiplayer won't meet the following requirements:

- Spaceships shall collide.