Requirements LaserSharks

by

Group 27

TI2206 Software Engineering Methods of the Computer Science curriculum at the Delft University of Technology.

Sytze Andringa (sytze@morvigor.nl)
Youri Arkesteijn (youri.essa@gmail.com)
Stefan Breetveld (spiderbiggen@gmail.com)
Michiel van den Berg (m.r.vdberg95@gmail.com)
Daan van der Werf (daanvanderwerf@hotmail.com)

Table of Contents

- Product Owner story
- Functional Requirements
  - Must haves
  - shall haves
  - Could haves
  - Wouldn’t/won’t haves
- Non-functional requirements

Product Owner story

“As The player, I want a game with a laser shark so that the laser shark could swim in the sea. To win the game, the laser shark shall eat other fishes. When the laser shark eats smaller fishes, it grows. When the laser shark tries to eat a bigger fish, it shall die. When the player is ready to play, he shall start and stop the game whenever he wants. The player wins the game when the laser shark is the biggest fish of the sea.” ~ The product owner.

Functional Requirements

Must haves

- The player shall open the game and see an empty board.
- The player shall start the game when the game has opened.
  This shall be done by pressing a start button.
- The player shall see his own shark when the game has started.
- The player shall see fishes move around when the game has started.
- The player shall see fishes of different sizes.
- The player shall let his shark move around the board.
  The player shall be able do so by pressing the arrow keys.
- The player shall collide with fishes by moving his shark into them.
- The player shall kill a smaller fish by moving his shark into them.
- The player shall grow his shark by having his shark collide with a smaller fish.
  The rate of growth is a standard amount per fish, independent of the size of the fish
- The player shall lose the game by having his shark collide into a bigger or equal sized fish.
- A fish shall disapear from the board when it’s killed.
- A fish shall move around on the board.
- The player shall win the game when his shark reaches a certain size.

Should haves

- The player shall start a new game of LaserShark when the previous one has ended.
- The player shall stop a game of LaserShark that is currently in progress.
- The game shall end when the player loses, dies or stops it.
- The game shall randomly assign a size to the fishes.
- The game shall initiate and show the player just above the minimal size.
- The game shall clear the board when the game has stopped.
- The fishes shall move controlled by the program from side to side.

Could haves

- The player shall have the option to pause the game while in progress.
- The game shall play a sound when his shark eats a fish.
- The game shall play a music theme when in progress.
- The player shal have the option to adjust the game’s resolution.
- The player shall have the option to mute the sound.
- The game shall initiate and show the player’s score at 0.
- The game shall reward the player with a certain amount of points for eating smaller fishes.
  This amount depends on the type of fish eaten and is different from the growth of the shark.
- A fish shall move away from the shark if the shark is larger than the fish.
- A fish shall move towards the shark if the shark is smaller than the fish.

- The game shall display a highscore.
- The player shall have the option to save and load a game.
- The player shall see a menu when opening the game.
- The player shall be able to change the difficulty

Wouldn’t/won’t haves

- The game shall show a timer showing the player how long the game has been running.
- The game shall have advertisements.

Non-functional requirements

- The game shall be playable on Windows (7 or higher), Mac OS X (10.8 and higher), and Linux.
- The game shall be implemented in Java 8.
- The project shall make use of Maven.
- Every implemented (non enum and GUI) class shall have at least one test class.
- The project shall have a minimum of 75% branch coverage (for non Enum and GUI classes).
- All code shall be programmed according to our standard checkstyle rules.
- A first fully working version of the game shalll be delivered at September 11, 2015.
- For the iterations after the delivery of the first fully working version, the Scrum methodology shall be applied.
- The implementation of the game shall have at least 75% line test coverage (for non Enum and GUI classes).
- One team member shall be the scrum master.
  The scrum master is responsible that scrum is used properly. He keeps overview of the sprints and leads the meetings.
- One team member who is not the scrum master shall be the product owner.
  The product owner represents the costumer. He tells the rest of the team what he would like the product to become.
- The team shall have group meetings at least once a week.
  The meetings will be on EWI. Every meeting we will discuss what has been done in the previous sprint and what we will do the next sprint.