Fishy Requirements

TI2206 Software Engineering Methods (2015-2016 Q1)
Date: 02/09/2015
Group Number: 0
Group Members:

- Thomas Baars - tcbaars@gmail.com
- Lars Ysla - yslars@gmail.com
- Boris Schrijver - boris@radialcontext.nl
- Adriaan de Vos - adriaan.devos@gmail.com
- Dylan Straub - d.r.straub@student.tudelft.nl

Table of Contents

1. Functional Requirements

For the game Fishy, the requirements which describe the systems services and functions have been grouped under the functional requirements. These functional requirements have then been divided into four categories based on the MoSCow model for prioritizing requirements; and are as follows:

1.1 Must Haves

- The game must show an empty screen, with only the user's fish at the centre, when a new game starts.
- Upon initiation of the game, the user's score must be set to 0.
- The user must be able to control the direction in which the fish moves (via arrow keys).
- There must be other fish of various (randomly generated) sizes, swimming either left-to-right or right-to-left.
- The user's fish must not be able to leave the playing field.
- The other fish fish must be spawned at either the left or the right side of the screen, such that they appear to be swimming into the playing field.
- The fish must have staggered spawns, and there must be a maximum and minimum number of fish spawned, at a time, such that the user has space to move.
- A fish is considered eaten when the a fish collides with a smaller fish.
- The spawned fish must not collide with / eat each other.
- Each fish eaten by the user must increase the user's score.
- Each fish eaten by the user must increase the size of the user's fish.
- The game is considered lost when the user's fish is eaten.
- There must be a limit to how large the user's fish can become.
- Once the user's fish reaches the size limit no more fish spawn, and when the last fish is eaten the user wins, and the game is over.

1.2 Should Haves

- The user's fish should face the direction in which it is swimming.
- The amount by which the score is incremented should be proportional to the size of the fish that is eaten.
- The current score should be displayed in-game.
- When the game is launched, there should be a menu screen from which the buttons "Play", "Instructions" and "Exit" can be pressed.
- There should be a way to view instructions on how to play the game.
- There should be a "Game Over" screen, when the game is over, with the achieved score and an option to play again.
- The user should be able to pause and resume the game.
- The user should be able to quit the game, while in-game.

1.3 Could Haves

- The game could have bubbles and sea plants to improve the appearance of being underwater.
- The game could have a background image that adds to the appearance of being underwater.
- The fish could have tails which move, to simulate the appearance of swimming.
- The interaction by which a fish is eaten should depend upon the orientation of the fish which is doing the eating (e.g. it would be preferable if a fish cannot eat another fish through its tail as opposed to its mouth).
- There should be a progress indicator for how many fish the user's fish has eaten (e.g. a tally that uses different size fish skeletons to represent the current count).
- The game should play a music theme.
- The music should be able to be toggled on and off in-game.
- A sound should play when a fish is eaten.
- A sound should play when the user wins.
- A sound should play when the user loses.
- A sound should play when a menu button is pressed.
- The sound should be able to be toggled on and off in-game.
- The user's fish should decrease in size if it has not eaten a fish within a certain amount of time.
- The amount of time it has to eat a fish before it decreases in size should be shown on screen, e.g. a hunger bar or a countdown.
- The user's fish will not decrease in size once it has reached its starting size.
- The fish should have various sprites.
- The top ten high scores should be saved.
- The current high score should be displayed in-game.
- The high scores should be viewable from the starting menu and "Game Over" screen.
- The user should be able to enter a nickname to associate with the achieved score, once the game is over.
- A nickname should be recorded with the associated high score.

1.4 Would/Won't Haves

- The game should have an online multiplayer mode, where another player can control a second fish in the same playing field.
- The game should have a "Dolphin mode". In which the user's "dolphin" has a "lung capacity bar" which depletes over time; and the "dolphin" must therefore surface for air to refill it. If the "lung capacity bar" is completely depleted the "dolphin" suffocates and it is considered game over.
- The game should have intelligent fish. This includes:
    - "Sharks" which actively follow the user's fish.

  - Fish that that do not only go left-to-right or right-to-left but might change directions.
  - Some smaller fish will be afraid of the user’s fish and actively try to avoid it.
- The game should have additional obstacles, such as fishing nets, pollution spots, fishing hooks and sea mines. These additional obstacles should result in a game over if the user’s fish collides with them.
- The game should have power-ups, such as invincibility and increased speed.
- The game should have bonus items, such as pearls, which add to the user’s score.
- The game should have achievements, such as “beat the game” and “eat 100 fish in one game”.
- The game should have the option to play the game with the mouse instead of the keyboard.
- The game should increase in difficulty as the game progress, which means that the spawned fish move at increased speeds, and more fish are spawned.
- The game should have the option to change the initial difficulty of the game.
- The game should have the option to customize the user’s fish before starting a new game.

2. Non-Functional Requirements

For the game Fishy, the requirements which outline the constraints, on the system and the developments process, set forward by our stakeholders have been grouped under non-functional requirements; and are as follows:

- The game should be playable on Windows 7 (and higher), Mac Os X (and higher), and Linux Ubuntu 14.04 (and higher).
- The game should be implemented in Java.
- Git should be used as a version control system, via GitHub.
- JUnit should be used to perform unit testing.
- Apache Maven should be used for project management.
- Travis should used as a continuous integration tool.
- PMD, CHeckstyle, and FindBugs should be used to manage source code.
- Octopull should be used to manage pull-requests.
- A first fully working version of the game should be delivered on Friday, 11 September 2015.
- For the iterations after the delivery of the first fully working version, the SCRUM methodology should be applied.
- The final product should be delivered on Friday, October 30 2015.
- For each iteration, excluding the first working version, the implementation of the game should have at least 75% of meaningful test coverage, where meaningful means that the test are actually testing the functionality of the game and for example not just executing the methods involved.