Fishy Requirements

Informational links:
Gameplay video: https://www.youtube.com/watch?v=G3vC6x7CTSIh
Flash game: http://spele.nl/fishy-1-spel/
Modern spin-off: http://agar.io/

Functional Requirements

Must Haves

- The game must show an empty screen, with only the user's fish at the centre, when a new game starts.
- User must be able to control the direction in which the fish swims (via arrow keys).
- There must be other fish of various (randomly generated) sizes, swimming either left-to-right or right-to-left.
- The fish must have staggered spawns.
- 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 eaten when the user's fish collided with a smaller fish.
- Each fish eaten must contribute to the user's score.
- Each fish eaten must contribute to the fish's (increased) size.
- The game must end when the user's fish is eaten.
- The user's fish is eaten if it collides with a larger fish.
- There must be a limit to how large the user's fish can become.
- Once the user's fish reaches the limit no more fish spawn, and when the last fish is eaten the user wins.
- The other fish must not collide with / eat each other.
- The user's fish must not be able to leave the screen.
- Upon initiation of the game, the user's score must be set to 0.

Should Haves

- 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", "instruction" and "quit" can be pressed.
- There should instructions on how to play the game.
- There should be a "Game Over" screen with an option to play again and the achieved score.
- The user should be able to pause the game.
- The user should be able to quit the game.
- The user's fish should face the direction in which it is swimming.

Could Haves

- The game could have bubbles and sea plants to improve 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 could 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 could be a progress indicator for how many fish the user's fish has eaten (e.g. in the form of fish skeletons or the like).
- Could have music play on a loop.
- 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.
- Hunger bar: Player fish will decrease in size if he hasn't eaten for a while
- Various sprites for the wild fish
- The top ten high scores could be saved.
- The current high score could be displayed in-game.
- The high scores could be viewable from the starting and/or "game over" screen.
- A nickname could be recorded with the associated score.

Would/Won't Haves
- Multiplayer
- "Dolphin mode": in which the "fish" (dolphin) must occasionally surface for air
- Smart AI fishes (sharks that follow the player, fishes that don't just go left-to-right or right-to-left but might change direction)
- Additional obstacles (fishing nets, pollution)
- Upgŕades (eg. invincibility, increased speed, no hunger)
- Additional bonus points (Pearls or money dropping in the water)
- Achievements (eg. Beat the game on medium, eat 250 fish in one game)
- Option to play the game with the mouse instead of the keyboard
- Campaign mode with different levels of increasing difficulty and different fish/challenges
- Player fish customization
- Small wild fish will be scared of the player fish and try to turn around/move away
- Various difficulty levels
- Threats or dangerous events to the player fish (eg. Water mines, fishing hooks, Big sharks)

Non-Functional Requirements
- 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 8.

- 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 be the continuous integration tool used.
- 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.
- The first working version will have unit tests to cover the basic functionality.