Software Engineering Methods Project

TI2206 Bubble Trouble Requirements

Group 10

Faris Elghlan
4341538
faris.elghlan@hotmail.com

Marco Houtman
4307232
M.Houtman@student.tudelft.nl

Bastijn Kostense
4372972
bastijnkostense@hotmail.com

Joshua Slik
4393007
JoshCode@protonmail.com

Lisette Veldkamp
4394712
L.S.Veldkamp@student.tudelft.nl

Maiko Goudriaan
Teaching Assistant
maiko.goudr@gmail.com

TU Delft

This page is intentionally left blank.

Chapter 1

Requirements

1.1 Functional Requirements

We will use the MoSCoW method to define the functional requirements for the game Bubble Trouble.

1.1.1 Must Haves

- The game shall start on level 1 with a playable character and at least one ball
- The playable character shall be visualized as a picture to be determined at a later date, either animated or static.
- The playable character shall have 5 lives at the start of the game
- The playable character shall be able to stand still, move to the left and to the right and fire his water gun upwards
- The playable character shall be controlled by the player using keys on the keyboard
- The playable character shall be able to use his water gun to shoot the balls in the level
- The projectile that is shot from the water gun shall leave a trail and shall move upwards
- The projectile and its trail shall disappear, when it collides with the ceiling
- The balls shall be placed at different heights
- The balls shall be visualized as circles in different colors
- The minimum height at which the balls shall be placed will make it able for the player to walk under it when the ball is at its maximum height.
- The maximum height at which the ball shall be placed will make it able that if the player shoots it at the maximum height, the two smaller balls will hit the ceiling
- The balls shall bounce across the level
- When a ball bounces against a wall, it shall bounce in the direction it came from
- When a ball which is not the smallest size possible is hit, it shall split in two other, equally sized, smaller balls
- When a ball that is not the smallest size possible gets hit by the projectile from a water gun or its trail, one resulting ball shall go upwards and to the left, and the other shall go upwards and to the right
- When a ball with the smallest size possible gets hit, it shall disappear
- When all the balls in a level have disappeared, the level shall be completed

- The ceiling in the level shall have spikes, and when a ball hits a spike, the ball shall disappear
- The game shall have a score counter
- The score of the player will be increased when shootings balls and completing levels
- A level shall grant a certain amount of points based on the time left
- When a ball hits the playable character, the player shall lose a life and the level shall get restarted
- The game has a timer, if the timer runs out, the player shall lose a life and the level shall get restarted
- The timer will get reset, when a new level is loaded
- The game, when the player is out of lives, shall be over

1.1.2 Should Haves

- The game shall have currency dropped by destroying the balls which the player can pick up and will increase their score
- The game shall have power-ups dropped by shooting the balls which the player can pick up and will change the properties of the gun
- The balls shall have a different pattern in which they bounce depending on their size
- The game shall have different sorts of balls, which differ in bouncing pattern
- The game shall have a main menu with an option to start the game.
- The game, when all levels are completed or all lives are depleted, shall show a game over screen, showing a button to return to the main menu and the score achieved by the player

1.1.3 Could Haves

- In certain levels, the game shall have a downwards moving ceiling
- In certain levels, the game shall have doors which the player can pass through when a certain amount of balls is cleared from the level
- The game shall have a high scores button on the main menu, where players can view the high scores which have been achieved
- The game shall have an option to skip levels which have already been completed
- The game shall have background music and/or sound effects
- The main menu shall be extended with the option to choose previously won levels; a button to enable/disable music; sound effects and change controls.

1.1.4 Wont Haves

- The ability for more than two players to play together simultaneously.

1.2 Non-Functional Requirements

- The game shall be playable on Microsoft Windows 7 and later versions
- The game shall be implemented in Java
- A first fully working version of the game shall be delivered September 11th, 2015
- After the delivery of the first fully working game, there shall be weekly iterations applying the scrum methodology
- The final working version of the game shall be delivered October 30th, 2015
- Git shall be used during this project as version management tool
- GitHub shall be used during the project to host our root Git repository
- Apache's Maven shall be used as a tool to have a completely reproducable and automated compilation pipeline; together with the folowing plugins:
  - Cobertura shall be used during the project to check the project's test coverage
  - FindBugs, PMD and Checkstyle shall be used during the project for static code analysis
- Travis CI and Octopull shall be used during the project for continuous integration.

1.3 Logger Requirements

- The logger wil be saved in log.txt.
- if there is no log file there will be created a log.txt.
- Timestamps will be added to every event in YYYY-MM-DD HH:mm:ss:ms
- If you press a button this should be logged in the gamelog file as "Button is pressed".
- If there is an keyboard interaction this should be logged in the logfile as "Key is pressed" and "Key is released".
- If there is an interaction with the ball this should be logged.
- If there is an interaction with the player this should be logged.

1.4 Definition of Done

A feature is considered "done" if and only if all of the following points apply:

- All of the requirements for the feature are met
- The feature has been thoroughly tested:
  - at least 80% of the model/control code has to be unit tested with Junit. Code for the view/layout does not have to be tested with Junit.
  - the view/layout code is tested by executing the program. The product owner must agree to the look of the view/layout.
- All old features which at one point in time worked must still function correctly
- The code has been documented:
  - all methods, including parameters and exceptions, are described with jdoc comments
  - all classes are described briefly (2-4 lines) as well
  - all packages and classes are described with UML, important methods of classes are present in the UML as well

  - The documentation has been made available to all of the members of the team (either on github or dropbox).

- The developer of the code has commited the code to the github repository
- The code has been merged into the main project
- The customer is satisfied with the feature (in case the customer does not have time, or cannot be reached to confirm his, or her, satisfaction considering the feature, the feature will be considered done for the time being and a new ticket to improve/adjust the feature can be made if the customer later is dissatisfied with the feature)

When all of the above points are achieved, the ticket associated to the feature will be closed and marked as done during the scrum meeting. If a feature is considered infeasible, unnecessary, or for any other reason aborted before completion, all documentation/code should be removed before the closing the ticket to ensure a clean working environment. If a feature has to be removed after it has been completed, the team will decide what to do during the first scrum meeting after the decision has been made.