Bubble shooter

Bavdaz, Luka 4228561
Clark, Liam 4303423
Gmelig Meyling, Jan-Willem 4305167
Hoek, Leon 4021606
Smulders, Sam 4225007

October 7, 2014

1 Requirements analysis

The stakeholder wants a game like Frozen Bubble, with both the singleplayer and multiplayer modes. Frozen Bubble is a Bubble shooter game, in which players have to shoot colored bubbles with a bubble gun. The bubble gun can be aimed and then shoot a bubble of a specific color. When the shot bubble hits a group of bubbles (two or more) of the same colour, these bubbles will pop and the player will receive an amount of points. When the player fails to hit a bubble of the same colour multiple times, a new row of bubbles will be inserted at the top, pushing the other bubbles. When one of these bubbles reaches the bottom of the screen, or all bubbles are shot, the game is over, and the score will be saved into a highscore board.

In the single player mode, players will face various levels with a different layout of the bubbles.

In the multiplayer mode, each player has his own canvas with a bubble gun and the bubbles to shoot (split screen). The players have different key bindings to rotate and fire the bubble gun.

2 Functional requirements

Each requirement is identified by a requirement identifier. This identifier consists of one or more marks pointing priority and status. For the priority status we use the *MoSCoW Prioritisation* (Clegg, 2014) (`M` for must have, `S` for should have, `C` for could have, `W` for won't have) and a unique number. Requirements that require further concideration are marked with `TBD`.

- `M-113` The player should be able to start a game through the menu
- When the player starts the game...
    1. `M-114` The canvas is filled with bubbles of certain colours (for example: red, green and blue), the colours are distributed depending on the game mode
    2. `M-115` Bubbles in the canvas are snapped to the top of the canvas
- `M-116` The player should be able to aim the bubble cannon (`left` and `right`) using the mouse.
- `M-117` The interface should show the colours for the upcomming bubbles to shoot
- Behaviour for when a bubble is shot
    1. `M-118` When a bubble hits a border of the canvas, it bounces
    2. `M-119` When a bubble hits another bubble, it snaps to it
    3. `M-120` When a bubble hits a group of bubbles of the same color, these bubbles should pop
    4. `M-121` When a bubble gets isolated from any bubble at the top of the canvas, they pop
    5. `M-122` When bubbles pop, points should be awarded to the player
    6. `M-123` When you do not succeed in popping bubbles, the player gets a penalty
    7. `M-124` For each a to be determined number of penalties a new row of bubbles with the remaining colours is inserted at the top, and other bubbles are shifted down

8. M-125 When a row of bubbles reaches the bottom of the canvas, the game is over
9. M-126 When all bubbles have popped, the game is finished

- There are two different modes to play:

  1. M-127 Singleplayer mode
     (a) M-128 When a player finishes the game, a new level will appear
     (b) M-129 A new level is generated by distributing bubbles randomly within a grid
     (c) S-130 In singleplayer mode, a player should also have the option to face various levels predefined by a designer
     (d) S-131 When the player clicks the highscore button he should be able to see a list of highscores
     (e) S-132 When the game has ended, the score and name of the player should be saved to the highscores
  2. M-140 Multiplayer mode
     (a) M-141 In the multiplayer mode, each player has his own canvas with a bubble gun and the bubbles to shoot (split screen)
     (b) M-142 The players have different key bindings to rotate and fire the bubble gun
     (c) M-143 When a player shoots a bubble, a new bubble is inserted in the other players screen
     (d) M-144 In the multiplayer mode, the bubbles are distrubuted randomly
     (e) M-145 The player should be able to play against another player remotely using a network connection
  3. M-146 The user should be able to switch between these modes through a menu

- C-150 Beginners tutorial. The controls and the goal of the game are explained when a player selects a button in the menu

- M-160 Every once in a while a player should get a power-up bubble loaded up in my cannon which has special abilities.

  1. M-161 A joker bubble should pop with a bubble of any color. If it is adjacened to less than 3 bubbles, the joker receives the colour of the bubble it collided with
  2. M-162 A bomb bubble should pop all bubbles in a certain radius around it after colliding
  3. M-163 A stone bubble can't be popped directly but only by popping the bubbles that connect it to the top
  4. M-164 A drunk bubble should move in a more difficult to predict way, so the user will have more difficulty accurately aiming the bubble
  5. M-165 A sound bubble makes a sound when shot, when moving, on impact or when popped
  6. M-166 A reversebomb bubble should add bubbles in all the empty spots in a certain radius around it after colliding
  7. S-167 A power-up bubble can have more than one special ability.

- M-170 The game should have logging functionalty on which game events are logged

  The logger should have the following functionality:

  1. M-171 Show a timestamp at which the event happend
  2. M-172 Show a priority of the event (*DEBUG*, *INFO*, *WARN*, *ERROR*)
  3. M-173 Show in which class and at which line number the log is called
  4. M-174 Ability to pass objects to a format string. For example: `log("called on {}", this)`.
  5. M-174 Ability to pass throwables to the log function, which causes the Throwables stacktrace to be printed to the outputstream
  6. M-175 Append log to system outputstream
  7. C-176 Append log to a text file

The following events should be logged:

1. `M-181` When a bubble is shot, along with its direction
2. `M-182` When a bubble is popped, along with which type of bubble it is
3. `M-183` When a row of bubbles is inserted
4. `M-184` When a ammo bubble is created
5. `M-185` When points are awarded to the player
6. `M-186` When looking for a multiplayer room
7. `M-187` When connected to a multiplayer room
8. `M-188` When disconnected from a multiplayer room
9. `M-189` When an exception is thrown

3 Non-functional requirements

3.1 Product requirements

- `M-231` The game should be able to run on the desktop computers at the TU Delft
- `M-232` It shouldn't take a user more than five minutes to learn the basics of the game

3.2 Organizational requirements

This game should be developed using the Java programming language, using the Maven, Git, Devhub tools for Continuous Integration and revision management and JUnit for Test-Driven Development (TDD). Within two weeks a fully functional game should be delivered. We have to work in a team of five.

3.3 External requirements

There are no external requirements at this point of the project.

References

Dai Clegg. Moscow prioritisation, may 2014. URL `http://dsdm.org/content/10-moscow-prioritisation`.