Requirements Tetris

by

The TAs

To be used as a reference model for the lab work of

TI2206 Software Engineering Methods
of the Computer Science curriculum

at the Delft University of Technology.

Supervisor:	Dr. A. Bacchelli
Teaching Assistants:	F. C. A. Abcouwer
	R. van Bekkum
	M. M. Beller
	T. Boumans
	A. Ferouge
	J. E. Giesenberg

Contents

1 Functional Requirements **1**

1.1 Must Haves 1

1.2 Should Haves 2

1.3 Could Haves 2

1.4 Would/Won't Haves 3

2 Non-functional Requirements **5**

1

Functional Requirements

For the game Tetris, the requirements regarding functionality and service are grouped under the *Functional Requirements*. Within these functional requirements, four categories can be identified using the *MoSCoW* model[1] for prioritizing requirements:

1.1. Must Haves

- The game shall show an empty board before a new game starts
- The game's board shall consist of a grid with a width of 10 cells and a height of 24 cells.
- The game shall place a new current tetromino in the top-middle of the board when the game starts
- The player shall be able to move the current tetromino either a cell to the left, right, or down at a time
- The player shall be able to rotate the current tetromino by 90 degree units
- The game shall let the current tetromino fall down slowly (each time after a set amount of milliseconds)
- The game shall not allow the current tetromino to be moved (partly) outside of the board
- The game shall not allow the current tetromino to be rotated (partly) outside of the board
- The game shall not allow the current tetromino to be rotated to (partly) overlap with other blocks already fixed on the game's board
- The game shall fix the current tetromino when it falls or moves down to the bottom of the game's board
- The game shall fix the current tetromino when it falls or moves down onto one or more fixed blocks on the board
- The game shall not allow a tetromino and its blocks to be moved when they are fixed on the board
- The game shall clear one or more rows of blocks on the board when the row is completely filled with blocks (*i.e.*, a horizontal line of 10 blocks was formed)
- The game shall move down all blocks in the rows above a row that is being cleared
- The game shall place a new tetromino in the top-middle of the board after the current tetromino has been fixed on the board (after which this new tetromino becomes the current tetromino)
- The player shall lose when a new tetromino is initially placed on a position on the board that already has one or more blocks on it

[1] http://en.wikipedia.org/wiki/MoSCoW_method

- The game shall allow the following shapes for the tetrominos: Z-shape, S-shape, I-shape, T-shape, J-shape, L-shape and O-shape[2]
- The game shall choose a random shape for each newly placed tetromino
- The game shall randomly rotate a new tetromino before placing it on the board

1.2. SHOULD HAVES

- The player shall be able to start a new game of Tetris
- The player shall be able to stop a game of Tetris that is currently in progress
- The game shall end a game when the player loses the game or stops it
- The game shall assign a randomly chosen (bright) color to each tetromino
- The game shall initiate and show the player's score at 0
- The game shall have multiple levels, and the player starts at level number 1 when the game starts
- The game shall keep track of the player's score using the original Nintendo scoring system:
  - Clearing a single row adds the following to the player's score: 40 * (level number)
  - Clearing two rows at a time adds the following to the player's score: 100 * (level number)
  - Clearing three rows at a time adds the following to the player's score: 300 * (level number)
  - Clearing four rows at a time adds the following to the player's score: 1200 * (level number)
- The game shall keep track of the number of doubles(two rows at a time), triples (three rows at a time) and tetrises(four rows at a time) cleared.
- The game shall for each level show the number of rows to clear needed to advance to the next level
- The game shall speed up the time interval in which the current tetromino automatically falls down when the player advances to the next level
- The game shall show a preview of the next tetromino that will be placed on the top-middle of the board
- The game shall clear the board when the game has stopped
- The game shall show the player's game statistics after losing a game
- The game shall reset the player's score and other game statistics(current level, number of doubles, triples and tetrises cleared) when a game ends
- The game shall show a visible grid on the board

1.3. COULD HAVES

- The game shall play a music theme when in progress
- The game shall play a sound when moving or rotating a tetromino
- The player shall be able to turn the music and sounds of the game on or off
- The player shall be able to toggle the board grid's visibility
- The player shall be able to pause the game while in progress
- The game shall keep track of the number of hurdles (clearing two or three rows separated by one or two rows with gaps)

[2] The basic tetrominoes: http://tetris.wikia.com/wiki/Tetromino

- The game shall award the player with an additional bonus score for creating hurdles
- The game shall show a prospect of where the current tetromino would land by showing a transparent copy of the tetromino on that position
- The player shall be able to fix the current tetromino all the way down directly on the position where it would land

1.4. Would/Won't Haves

- The game shall save the scores of all of its players using a high-score list with the top ten of names and scores of players with the highest scores
- The player shall be able to change the board's background by choosing an image from his/her computer files
- The game shall offer another game mode with *sticky* gravity.[3]

[3] Sticky gravity: http://tetrisconcept.nl/wiki/Line$_$clear#Sticky

2

Non-functional Requirements

Besides the provided functionality and services, design constraints need to be included in the requirements specification as well. These requirements do not indicate *what* the system should do, but instead indicate the *constraints* that apply to the system or the development process of the system.

- 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
- A first fully working version of the game shall be delivered at September 13, 2014
- 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% of meaningful line test coverage (where meaningful means that the tests actually test the functionalities of the game and for example do not just execute the methods involved)