SE 3XA3: Test Plan
PROJECT TETRIS

Team #11, Team Tetris
Daniel Agostinho agostd
Anthony Chang changa7
Divya Sridhar sridhad

December 6, 2016

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
12/05/2016	0	Initial Draft, Template of report
12/06/2016	1	Updated sections
12/07/2016	2	Updated sections

1 Functional Requirements Evaluation

The functional requirements of PROJECT TETRIS are straightforward. Our primary objectives while writing the code in terms of functionailty was the various keyboard responses to the game. These include horizontal and vertical translation of tetrominos, clearing of rows, a proper help menu and pause game function, and an adequate scoring system.

2 Nonfunctional Requirements Evaluation

2.1 Usability

PROJECT TETRIS adheres to the original arcade game both in terms of game logic and visuals. Our goal was to replicate as accurately as possible the existing implentation of this game, along with it's easy-to-use interface and straightforward gameplay. In order to determine if our game was easy to play, understand, and was enjoyable, we surveyed several people between the ages of 18-22, and asked them to quantitatively rate the game on a scale from 1-10. Other questions we asked the surveyed audience include rating the game in comparison to the original for similarity (from a scale of 1-5), as well as asking them to rate the visual graphics of the game from a scale of 1-10. The number of levels that a user was able to get to was also recorded, along with their high score.

2.2 Performance

Performance was measured and tested by running our game on multiple platforms, to ensure portability. PROJECT TETRIS was able to run on any machine containing a JVM, which includes Mac OS users, Windows users, and Linux users. Performance was also measured by testing whether or not there were any glitches while playing the game (i.e. if it ever crashed unexplicably), and also checking for lag times while running the executable .jar file.

2.3 Look and Feel

Look and Feel was also quantatively measured through survey responses. Users were asked to rate the appearance of the game from a scale of 1-10,

and were also asked to rate it in comparison to the original game of Tetris that they were familiar with.

3 Comparison to Existing Implementation

PROJECT TETRIS is based off of an existing open-source version of Tetris. Our goal was to maintain the look and feel as well as functionality of the original game, and adhere to its structure as much as possible. In comparison to the existing implementation, PROJECT TETRIS has few changes. However, there is added functionality in the form of a help menu, which appears at any point throughout the game when the user presses the 'h' key. Other additional features include levels, which increase as the number of rows cleared in the game increases. With each subsequent level, the constant speed of the tetrominos moving downward increases as well, thereby making gameplay more challenging. Scoring is also unique, as there are points given based on how many rows are cleared and how quickly they were cleared as well (i.e. depending on whether the spacebar was used to accelerate tetrominos downward faster).

4 Unit Testing

Unit Testing was very succesful and useful in determining whether all functional requirements were met. JUnit test classes was used, as this was a Java project. This also enabled us to automate test cases to determine the basic functionality of the game.

5 Changes Due to Testing

There were no changes made after unit testing, as it showed that all the functional requirements were met.

6 Automated Testing

Unit testing using JUnit enabled us to automatically determine if any one of our functional requirements were not being met. If issues arose in the code

and we were uncertain of where they were, unit testing allowed us to automatically test step-by-step and debug the code. The automated tests that were conducted tested to make sure the tetrominos responded to keyboard input, as well as performed other important functions such as clearing of rows, and increasing of levels/proper scoring.

7 Trace to Requirements

Please refer to Table 2.

8 Trace to Modules

Please refer to Table 3.

9 Code Coverage Metrics

During automated testing, we ensured that all code was tested to determine whether both the game logic and the outward visuals were updating properly. Coverage criteria included Function coverage, Condition coverage, Branch coverage, and Statement coverage. All edge cases were tested in the form of boolean checks for each step of the code, to ensure that it was executed properly. Code coverage was therefore 90%.

Table 2: **Trace between Requirements and Tests**

Requirement	**Test**
Horizontal translation	Press right arrow, tetromino moves right
Horizontal translation	Press left arrow, tetromino moves left
Vertical translation	Press spacebar, tetromino moves down at faster rate
No key input	Tetromino moves down at constant rate
Clearing rows	Row disappears once tetrominos are in well
Help menu	Press 'h' key, instructions appear in new window
Pause function	Press 'p' key, game pauses and tetrominos stop moving
Increase level	Clear 10 rows, level goes up and speed increases
New game	Once end game condition reached, new game can start

Table 3: **Trace between Requirements and Tests**

Module	**Test**
TetrisController	Unit tests to check if game logic worked
TetrisController	Unit tests to check if keyboard input received
Piece	Unit tests to check if states were updated
Well	Unit test to check if state was updated
Painter	Visual test to see if elements were drawn