# Test Plan

# Super Tetris

Group#: 38

Team Name: Binary

Members:

Tongfei Wang : wangt62

Bowen Yuan : yuanb1

Tim Zhang : zhangj14

December 6, 2017

SFWR ENG 3XA3

McMaster University

# Contents

# List of Tables

# List of Figures

Table 1: **Revision History**

| **Date** | **Version** | **Notes** |
|---|---|---|
| Oct 27 2017 | 1.0 | General Info & Plan Uploaded |
| Oct 27 2017 | 1.1 | System Test Descriiption Uploaded |
| Oct 27 2017 | 1.2 | Tests for PoC Uploaded |
| Oct 27 2017 | 1.3 | Unit Testing Plan Uploaded |

# 1 General Information

## 1.1 Purpose

The purpose is to test Super Tetris which is re-implemented from the Tetris game. This document will make a test plan which shall have a testing structure. Following the structure, several test cases shall be able to generate . ~~and test both functional and non-functional requirements.~~ The test cases cover both funcitonal and non-functional requirements.

## 1.2 Scope

As the purpose is to test a software project, the tests shall meet both functional and non-functional requirements.
For functional requirement, ~~is~~ it shall test all classes and functions of the program including the original codes. In term of non-functional requirement, it shall test if the program meets the developing requirement and ~~all~~ if all the added elements works well. Also, the added icons, pictures, and ~~anime~~ animation shall appear at the right place ~~at~~ and a right time.

## 1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

| **Abbreviation** | **Definition** |
|---|---|
| POC | Proof of Concept |
| FR | Functional Requirement |
| NFR | Non-Functional Requirement |
| NFR | Non-Functional Requirement |
| SP | Super Tetris |
| IN | Input and Navigation |
| GL | Game Logic |

Table 3: **Table of Definitions**

| **Term** | **Definition** |
|---|---|
| Tetrominoes | Geometric shape composed of four squares |
| Item | Four different items to help player get higher scores |
| Line Clear | Theway to obtain score in Super Tetris |
| Score | Measure of Player's performance |

### 1.4 Overview of Document

~~In this document, it will make a plan for the testing.~~ This document shows the testing plan in detail.At first, the document will briefly introduce the software to be tested, test team and the testing tools. Secondly, according to the scope of testing, it will talk about the details about the implemented function to be tested and the way it works. The article will also clarify that how the test team overcomes the challenging testing. As the unit testing ~~shall be used~~ is used and it is essential part of our testing process ~~to introduce the plan to it at last~~,the introduction of it will be at the end of this document .

## 2 Plan

### 2.1 Software Description

Tetris is a tile-matching puzzle game. "Tetrominoes" are differently shaped blocks and they drop from the top of the screen one by one. The player can move or rotate the Tetriminos as they fall. The goal of this game is to create a horizontal line without gaps to gain the score of the game. ~~Once the line is created, it will disappear and be replaced by the blocks above it.~~

### 2.2 Test Team

The test team will consist of Tim Zhang, Bowen Yuan, Tongfei Wang. The three main testers will split the entire testing evenly covering all the different types of testing and preparing the automated testing.

## 2.3 Automated Testing Approach

Automated testing will be used for testing of the game mechanics system and it will utilize both unit testing and coverage analysis. To ensure that unit testing can be run while the game mechanics are in development the test cases have been grouped into sections that can be fully implemented once certain subsets of the game mechanics systems are completed.

## 2.4 Testing Tools

Super Tetris is programmed in Javascript, So we decide to use 'Mocha' to automate the unit testing.

## 2.5 Testing Schedule

| **Task** | **Team Member** | **Date** |
|---|---|---|
| User Input | Bowen | Oct 30th 2017 |
| Navigation | Bowen | Nov 4th 2017 |
| Game Logic | Micho | Nov 4th 2017 |
| Usability | Tim | Nov 10th 2017 |
| Performance | Micho | Nov 10th 2017 |

Table 4: Testing Assignments

See Gantt Chart at the following url ...

# 3 System Test Description

## 3.1 Tests for Functional Requirements

### 3.1.1 User Input and Navigation Test

| | |
|---|---|
| **Test 3.1.1.1:** | **SP-IN1** |
| **Type:** | Functional,Dynamic,Manual |
| **Initial State:** | Web browser is open |
| **Input:** | User enters the domain of this game |
| **Output:** | A new game starts |
| **Pass:** | We will use different ~~mobile devices and PC~~ browsers to check if the game interface is displayed successfully after entering our game domain ~~in web browser~~. |

| | |
|---|---|
| **Test 3.1.1.2:** | **SP-IN2** |
| **Type:** | Functional,Dynamic,Manual |
| **Initial State:** | Game is in process |
| **Input:** | User presses 'enter' |
| **Output:** | Game paused |
| **Pass:** | We will check if game is paused after pressing 'enter' when game is in process. |

| | |
|---|---|
| **Test 3.1.1.3:** | **SP-IN3** |
| **Type:** | Functional,Dynamic,Manual |
| **Initial State:** | Game is paused |
| **Input:** | User presses 'enter' |
| **Output:** | Game resumes |
| **Pass:** | We will check if game is resumed successfuly after pressing 'enter' when game is paused |

| | |
|---|---|
| **Test 3.1.1.4:** | **SP-IN4** |
| **Type:** | Functional,Dynamic,Manual |
| **Initial State:** | Game is paused or in process |
| **Input:** | User presses 'r' |
| **Output:** | Game is restarted |
| **Pass:** | We will check if game is restarted successfuly after pressing 'r' when game is paused or in process |

| | |
|---|---|
| **Test 3.1.1.5:** | **SP-IN5** |
| **Type:** | Functional,Dynamic,Manual |
| **Initial State:** | Game is in process |
| **Input:** | User presses ~~'up'~~ 'w' |
| **Output:** | The current tetromino is rotated clockwise |
| **Pass:** | We will check if the current tetromino is rotated clockwise when user presses 'up' |

| | |
|---|---|
| **Test 3.1.1.6:** | **SP-IN6** |
| **Type:** | Functional,Dynamic,Manual |
| **Initial State:** | Game is in process |
| **Input:** | User presses 'left' or 'right' by one grid |
| **Output:** | The current tetromino is moved left/right |
| **Pass:** | We will check if the current tetromino is moved left/right by exactly one gird when user presses 'left' or 'right'. |

**Test 3.1.1.7:** **SP-IN7**

**Type:** Functional,Dynamic,Manual

**Initial State:** Game is in process

**Input:** User presses ’down

**Output:** The current tetromino falls to the bottom(not intersected with other tetrominoes)

**Pass:** We will check if the current tetromino falls to the bottom when user presses ’down’

**Test 3.1.1.8:** **SP-IN8**

**Type:** Functional,Dynamic,Manual

**Initial State:** Game is in process

**Input:** User presses ’1’ or ’2’ or ’3’ ~~or 4~~

**Output:** User buys and uses the item they selected(’1’ to ~~’4’~~ ’3’ correspanding to four different items)

**Pass:** We will check if users can use the item they selected successfully when they press ’1’ to ~~’4’~~ ’3’

### 3.1.2 Game Logic Test

| | |
|---|---|
| **Test 3.1.2.1:** | ~~**SP-GL1**~~ |
| **Type:** | ~~Structrual,Dynamic,Automated~~ |
| **Initial State:** | ~~Game is in Process.~~ |
| **Input:** | ~~a 4*4 (represent a certain tetromino)~~ |
| **Output:** | ~~a 4*4matrix(represent an input tetromino rotating 90 degrees clockwise)~~ |
| **Pass:** | ~~We will use assert to check if function rotateRight matrix) can accept a 4*4 metrix and return a 4*4 matrix rotating 90 degrees clockwise.~~ |

| | |
|---|---|
| **Test 3.1.2.2:** | **SP-GL2** |
| **Type:** | Structrual,Dynamic,Automated |
| **Initial State:** | Game is in Process. |
| **Input:** | Coordinate of current tetromino which is a 4*4 ~~metrix~~ matrix |
| **Output:** | Boolean |
| **Pass:** | We will use assert to check if function intersects can judge the current tetromino intersects the border or other tetrominoes. |

| | |
|---|---|
| **Test 3.1.2.3:** | **SP-GL3** |
| **Type:** | Structrual,Dynamic,Automated |
| **Initial State:** | Game is in Process. |
| **Input:** | row number |
| **Output:** | new row or number of rows killed |
| **Pass:** | We will use assert to check if funcion killrows() can kill a(some) filled row and increase the number of rows killed |

| | |
|---|---|
| **Test 3.1.2.4:** | **SP-GL4** |
| **Type:** | Structrual,Dynamic,Automated |
| **Initial State:** | Game is in Process. |
| **Input:** | N/A |
| **Output:** | A random 4*4 piece(tetromino) |
| **Pass:** | We will use assert to check if function randomPiece() can return a 4*4 ~~metrix~~ matrix (any tetromino) randomly. |

| | |
|---|---|
| **Test 3.1.2.5:** | ~~**SP-GL5**~~ |
| **Type:** | ~~Structrual,Dynamic,Automated~~ |
| **Initial State:** | ~~Game is in Process.~~ |
| **Input:** | ~~N/A~~ |
| **Output:** | ~~A playfield can be drawed~~ |
| **Pass:** | ~~We will use assert to check if function newGame() can create a 20*10 playfield successfully~~. |

| | |
|---|---|
| **Test 3.1.2.6:** | **~~SP-GL6~~** |
| **Type:** | ~~Structrual,Dynamic,Automated~~ |
| **Initial State:** | ~~Game is in Process.~~ |
| **Input:** | ~~Number of rows killed~~ |
| **Output:** | ~~Experience~~ |
| **Pass:** | ~~We will use assert to check if function getExp() can return the number of Experience which Experience = 10*number of rows killed~~ |

| | |
|---|---|
| **Test 3.1.2.7:** | **SP-GL7** |
| **Type:** | Structrual,Dynamic,Automated |
| **Initial State:** | Game is in Process. |
| **Input:** | item |
| **Output:** | boolean |
| **Pass:** | We will use assert to check if function haveEnoughMoney() can return whether a user can buy the item he selected |

**Test 3.1.2.8:** **~~SP-GL8~~**

**Type:** ~~Structrual,Dynamic,Automated~~

**Initial State:** ~~Game is in Process.~~

**Input:** ~~price of item~~

**Output:** ~~Experience~~

**Pass:** ~~We will use assert to check if function newExp() can return whether the remaining Exp when user bought certain item.~~

**Test 3.1.2.9:** **SP-GL9**

**Type:** Structrual,Dynamic,Automated

**Initial State:** Game is in Process.

**Input:** Number of rows killedplaying time

**Output:** Score

**Pass:** We will use assert to check if function getScore() can return the number of Score which Score $\bar{1}$00*number of rows killed + 10*time(second)

**Test 3.1.2.10:** **SP-GL10**

**Type:** Structrual,Dynamic,Automated

**Initial State:** Game is in Process.

**Input:** N/A

**Output:** speed of falling

**Pass:** We will use assert to check if function useSlow() can return the speed of falling which is half of original speed and lasts 30 seconds.

**Test 3.1.2.11:** **SP-GL11**

**Type:** Structrual,Dynamic,Automated

**Initial State:** Game is in Process.

**Input:** N/A

**Output:** 4*4 ~~metrix~~ matrix

**Pass:** We will use assert to check if function useSkip() can return a new random 4*4 ~~metrix~~ matrix which override the original ~~metrix~~ matrix.

**Test 3.1.2.12:** **SP-GL12**

**Type:** Structrual,Dynamic,Automated

**Initial State:** Game is in Process.

**Input:** N/A

**Output:** rows or number of row killed

**Pass:** We will use assert to check if function useClear() can kill the top row and return the new rows and ~~increse~~ increase the number of rows killed.

**Test 3.1.2.13:** **SP-GL13**

**Type:** Structrual,Dynamic,Automated

**Initial State:** Game is in Process.

**Input:** Playfield

**Output:** Boolean

**Pass:** We will use assert to check if function gameover() can return the game is over or not.(When part of a new tetromino(or the whole part) cannot be displayed in game playfield because of collision with other tetrominoes that game is over)

| | |
|---|---|
| **Test 3.1.2.14:** | **SP-GL14** |
| **Type:** | Structrual,Dynamic,Automated |
| **Initial State:** | State: Game is paused |
| **Input:** | N/A |
| **Output:** | Exception |
| **Pass:** | When user tries to move the tetromino when game is paused then a exception will be raised. |

## 3.2 Tests for Nonfunctional Requirements

### 3.2.1 Performance

| | |
|---|---|
| **Test 3.2.1.1:** | **Precision** |
| **Description:** | The event shall happen at the correct time. |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | The boom shall explode when it stops. The ice shall slow the falling time when it is triggered. The line shall be cleared when a complete line is made. |

| | |
|---|---|
| **Test 3.2.1.2:** | **Capacity** |
| **Description:** | Super Tetris needs less than 10MB in memory. |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | Test team calculates the size of whole program file before uploading. |

### 3.2.2 Usability

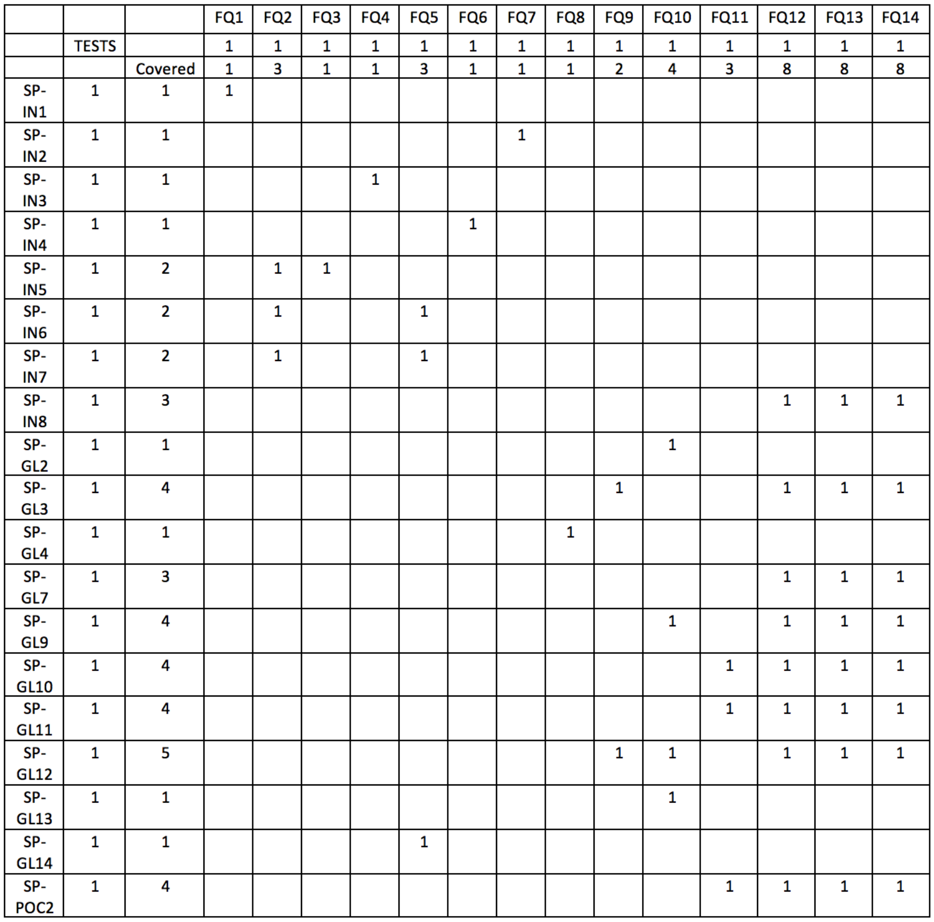
| | |
|---|---|
| **Test 3.2.2.1:** | **Operating System Support** |
| **Description:** | Super Tetris shall run on all major web browsers ( Chrome/Firefox/Safari /IE) |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | Test team run and play the game through Chrome/Firefox/Safari /IE. |

| | |
|---|---|
| **Test 3.2.2.2:** | **Look and Feel** |
| **Description:** | The falling of the tetrimino and line clear requires animation |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | When tetriminos are falling or when a row is eliminated, the animation display correctly. |

| | |
|---|---|
| **Test 3.2.2.3:** | **Difficulty** |
| **Description:** | Super Tetris shall follow the same rule in the Classic Tetris. |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | Average survey score of less than 4 in 'Difficulty' section. |

## 3.3 Traceability Between Test Cases and Requirements

All the failure test cases shall throw exception which shows where the issue from.



| | | | FQ1 | FQ2 | FQ3 | FQ4 | FQ5 | FQ6 | FQ7 | FQ8 | FQ9 | FQ10 | FQ11 | FQ12 | FQ13 | FQ14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TESTS | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | Covered | 1 | 3 | 1 | 1 | 3 | 1 | 1 | 1 | 2 | 4 | 3 | 8 | 8 | 8 |
| SP-IN1 | 1 | 1 | 1 | | | | | | | | | | | | | |
| SP-IN2 | 1 | 1 | | | | | | | 1 | | | | | | | |
| SP-IN3 | 1 | 1 | | | | 1 | | | | | | | | | | |
| SP-IN4 | 1 | 1 | | | | | | 1 | | | | | | | | |
| SP-IN5 | 1 | 2 | | 1 | 1 | | | | | | | | | | | |
| SP-IN6 | 1 | 2 | | 1 | | | 1 | | | | | | | | | |
| SP-IN7 | 1 | 2 | | 1 | | | 1 | | | | | | | | | |
| SP-IN8 | 1 | 3 | | | | | | | | | | | | 1 | 1 | 1 |
| SP-GL2 | 1 | 1 | | | | | | | | | | 1 | | | | |
| SP-GL3 | 1 | 4 | | | | | | | | | 1 | | | 1 | 1 | 1 |
| SP-GL4 | 1 | 1 | | | | | | | | 1 | | | | | | |
| SP-GL7 | 1 | 3 | | | | | | | | | | | | 1 | 1 | 1 |
| SP-GL9 | 1 | 4 | | | | | | | | | | 1 | | 1 | 1 | 1 |
| SP-GL10 | 1 | 4 | | | | | | | | | | | 1 | 1 | 1 | 1 |
| SP-GL11 | 1 | 4 | | | | | | | | | | | 1 | 1 | 1 | 1 |
| SP-GL12 | 1 | 5 | | | | | | | | | 1 | 1 | | 1 | 1 | 1 |
| SP-GL13 | 1 | 1 | | | | | | | | | | 1 | | | | |
| SP-GL14 | 1 | 1 | | | | | 1 | | | | | | | | | |
| SP-POC2 | 1 | 4 | | | | | | | | | | | 1 | 1 | 1 | 1 |

Figure 1: Traceability Testing Matrix

## 4 Tests for Proof of Concept

~~REDRAW TEST~~

| | |
|---|---|
| **Test 4.1:** | ~~**SP-POC1**~~ |
| **Type:** | ~~Dynamic,Manual~~ |
| **Initial State:** | ~~Game over~~ |
| **Input:** | ~~User presses 'r' or refresh the browser~~ |
| **Output:** | ~~New playfield replaces the original playfield and empty the score and Exp~~. |
| **Pass:** | ~~Press 'r' or refresh the website~~ |

BOMB TEST

| | |
|---|---|
| **Test 4.2:** | **SP-POC2** |
| **Type:** | Dynamic,Manual |
| **Initial State:** | Game is in process |
| **Input:** | user presses '1' |
| **Output:** | Bomb is used |
| **Pass:** | Press '1' and use a bomb to test to check if the 5*5 area around the bomb has been cleared up. |

The tests of Proof of Concept, we will use test SP-IN8, SP-GL7, SP-GL9, SP-GL10 as we describe above

## 5 Comparison to Existing Implementation

There shall be a separate package called testing in the program. The package shall include several classes with different testing methods. The classed shall invoke the methods and functions implemented without changing them. That is to say, all the implementation shall not be affected and remain unchanged after the testing.

# 6 Unit Testing Plan

Mocha shall be used as tool for unit testing.

## 6.1 Unit testing of internal functions

In order to test the internal functions, a unit test will be used to test most of the classes and methods implemented. We will generate a sequence of moves as input values. The sequence shall cover most implemented functions. The unit test shall use calculate right output values by math or logic. After that, it shall check whether the outputs match the right ones. It will throw exceptions if the program outputs a wrong value. Finally, a test coverage metrics will be used so as to calculate the how much percentage does the unit test include. At least 75 percent of the functions should be included.

## 6.2 Unit testing of output files

As the testing going, the interface of the game will change accordingly. The test will set several testing points in order to check if the ~~anime~~ animation and graphs go well. It shall take one screenshot of the interface per point. Also, as the test going, it shall generate the graphs according to the methods used as right outputs. Then, the similarity of both outputs shall be checked. At least a similarity of 98 percent shall be considered acceptable. For the cases below 98 percent, it will throw exceptions.

# 7 Appendix

This is where you can place additional information.

## 7.1 Symbolic Parameters

## 7.2 Usability Survey Questions

## User Experience Survey

The purpose of this survey is to help improving the game called Super Tetris. After playing the game, the players will be asked several questions about their playing experience.

Please select how do you think in the following areas:

**Entertainment:** 0 1 2 3 4 5
[ 0 = most boring, 5 = most fun ]

**Understanding:** 0 1 2 3 4 5
[ 0 = easy to understand, 5 = No idea ]

**Difficulty:** 0 1 2 3 4 5
[ 0 = easiest, 5 = most difficult ]

**Controls:** 0 1 2 3 4 5
[ 0 = non-intuitive, 5 = intuitive ]

**Interface:** 0 1 2 3 4 5
[ 0 = Clean and Nice, 5 = Complicated ]

Please write down what you think to the following questions:

1. Which is the best part do you think about the game.

2. Which part do you think it should be improved most.

3. The overall impression of the game.