SE 3XA3: Test Plan
Tetrileet

Team 15, AAA Solutions
Abdallah Taha, tahaa8
Ali Tabar, sahraeia
Andrew Carvalino, carvalia

March 4, 2021

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

This document ...

1 General Information

1.1 Purpose

The purpose of this document is to describe the testing tools, plan and actual test cases that will be used to verify Tetro Leagues' UI and back end system.

1.2 Scope

This test plan document will justify our intentions for testing the Tetrileet program. Our main objective is to encounter any existing game glitches or other defects, and later fix the program as necessary once they have been identified.
The test plan document is an approach to documenting all activities related to testing. Aspects of the software to be tested, testing tools used, and summaries of testing methods will all be presented.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
UI	User Interface
Abbreviation2	Definition2

Table 3: **Table of Definitions**

Term	**Definition**
Term1	Definition1
Term2	Definition2

1.4 Overview of Document

This document will cover our testing plan which will include the software description, test team, automated testing approach, testing tools, and testing schedule. It will also cover the system test description which will include the tests for functional and nonfunctional requirements. Furthermore, the tests for the proof of concept pertaining to Tetrileet will be provided. Finally, comparison tests of Tetrileet with another existing implementation of Tetris will be discussed, as well as a unit testing plan.

2 Plan

2.1 Software Description

The software is a simple remake of the classic video game known as *Tetris*. Randomized, preset shapes made up of blocks will fall from the top of a grid, able to be moved left or right in the grid by the player's input. The player can also rotate the blocks by 90, 180, 270, or 360 degrees. The blocks will keep falling until they reach the bottom of the grid, or land on top of another shape. If the player can arrange a full row of blocks by placing the shapes accordingly in the grid, that row of blocks will disappear, causing all blocks above to fall down one row. The player will be rewarded and gain points whenever this happens. If the top of the grid is reached by any blocks (the grid "overflows"), then the game is over.
Our software will feature a main menu in additional to the base game, as well as a difficulty adjusting feature - allowing users to change the speed of which the blocks fall.

2.2 Test Team

The test team for Tetrileet will consist of Abdallah Taha, Ali Tabar, and Andrew Carvalino.

2.3 Automated Testing Approach

The Jest testing library will be used to write automated tests for both the front-end and back-end of our program. For our UI, we will use the JavaScript testing libraries Jest and jsdom. These two libraries provide the functionality

to run tests on a mock DOM tree. They also allow us to select and manipulate DOM elements in order to verify some properties of our UI such as if an element has the correct height, width etc.

2.4 Testing Tools

The testing tools that we will use are the JavaScript testing libraries Jest and jsdom.

2.5 Testing Schedule

See Gantt Chart at the following url ...

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Hit Detection

Title for Test

1. test-id1

   Type: Functional, Dynamic, Manual

   Initial State:

   Input:

   Output:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

3.1.2 Area of Testing2

...

3.2 Tests for Nonfunctional Requirements

3.2.1 Area of Testing1

Title for Test

1. test-id1

   Type:
   Initial State:
   Input/Condition:
   Output/Result:
   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

3.2.2 Area of Testing2

...

3.3 Traceability Between Test Cases and Requirements

4 Tests for Proof of Concept

4.1 Area of Testing1

Title for Test

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

4.2 Area of Testing2

...

5 Comparison to Existing Implementation

6 Unit Testing Plan

6.1 Unit testing of internal functions

6.2 Unit testing of output files

References

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

7.2 Usability Survey Questions?

This is a section that would be appropriate for some teams.