Software Requirements Specification

for

Tetris

Version 2.0 approved

Prepared by Eddie Dounn
Ryan Dalan
Zachary Goldberg
Guillermo Gomez Jr.

Group SM2

5/06/2020

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
SRS CS4398	2/13/20	Initial SRS	1.0
SRS CS4398	5/06/20	Final SRS	2.0

1. Introduction

1.1 Purpose

This is a Software Requirement Specification (SRS) document for our build of Tetris. This document will describe the game environment, GUI, and backend. This document is intended as a reference for the game.

1.2 Product Scope

This SRS will describe the basics of operation and production of the Tetris game, including the backend, algorithm, GUI, and use cases. This is not intended to be an instruction manual for modifying the game, but rather for playing or extending the game. This SRS will not cover: Installing Java Runtime Environment or configuring a computer for use with this software.

1.3 References

- Team Version Control - https://github.com/edounn/cs4398-sm2
- Tutorial https://www.youtube.com/playlist?list=PLHwsL1JI79K2Kw5wf1KOOCn5KIXZ9qybv
- Java Swing - https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html
- Java Threads - https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html

2. Overall Description

2.1 Product Perspective

This product is a modern follow up of the original game of Tetris created in 1984. The product will contain the basic functionality of the original puzzle game but will also include new and modern interface as well as a newer visual appearance in comparison of the earlier development of the game.

2.2 Product Functions

a. GRAPHICAL USER INTERFACE
   The GUI will allow the user to interact with the game. The user interface will be used to start new games, play the game, show the user scores and relevant data. The user interface will also be responsible for allowing the user to modify program settings, such as, toggling the audio off or on, changing the control scheme, and/or putting the application into full screen mode.
b. ALGORITHM
   The main component of the backend of the game is the algorithm. The game will be run on a two-dimensional array which will be updated with the current position of the Tetris pieces. The algorithm will work by checking if there is a connected piece for the falling piece to connect to, or if it is clear to fall. The algorithm will then, upon the piece landing, check if a

row was completed, and if so will add to the user's score and clear the row from the screen and the array.

User Classes and Characteristics
The sole user class in this product is the player. As the player, the user will have access to starting a new game, and options to tweak their experience. These options include theme, window size, and controls.

2.3 Operating Environment

The software will be run on a Java Virtual Machine, and thus will run on any computer which supports Java (Windows, Mac OSX, Unix based systems, etc.). The software will also be able to be ported to web or mobile in the future, due to the versatility of Java. The game will require the user to have a keyboard and/or mouse, in order to control the game.

2.4 Design and Implementation Constraints

The software is open-source, and can be found on GitHub, and will be available at no cost to users. Users will be required to install a current version of either Oracle Java JRE, or an open source variant of the Java Runtime Environment. For issues involving the Java Runtime Environment the user is to contact the source of their variant, and not the developers of this software.

2.5 Use Cases

2.5.1 Default Use Case

Name: Title Screen
Trigger : User interaction with game and high scores
Precondition : User starts application

1. While in the application the user selects <START >.
2. While in the application the user selects <HIGH SCORES>.

Name: Start
Trigger : User interaction to begin game
Precondition : User starts application and inside Title screen.

1. While in the application the user selects <START >.
2. Game play screen displays.

Name: High Scores
Trigger : User interaction to see high scores
Precondition : User starts application

1. While in the application the user selects < HIGH SCORES >.
2. Game high score list displays.
3. In high score screen user selects <BACK> returns to title screen

Name: Pause
Trigger : User interaction to pause game
Precondition : User in game

1. While in the application the user selects <PAUSE >.
2. Game play stops and displays indication that game is paused.
3. Continue resumes game play

Name: Exit
Trigger : User interaction to quit current game
Precondition : User at title screen or pause screen

1. While in the title screen or pause screen user selects < EXIT >.
2. Game is closed.

2.5.2 Power User Use Case

Name: Theme Customization
Trigger: User interaction with the game and the blocks.
Precondition: User starts the game.

1. While in the application the user selects <Theme>.
2. While in the <Theme> menu, the user tweaks the color of the blocks to their liking.

2.5.3 Options Use Case

Name: Option Tweaking
Trigger: User interaction with the game and its options.
Precondition: User starts the game.

1. While in the application the user selects <OPTIONS>.
2. While in the <OPTIONS> menu, the user tweaks the options of the game to their liking. This includes keybindings, resolution, and volume.

3. External Interface Requirements

3.1 User Interfaces

The game will provide a user-friendly menu that the user can traverse through the various settings, game modes, and other functionalities provided in the game. The game will use Java's provided interface libraries such as AWT, Swing, and JavaFX.

3.2 Hardware Interfaces

The game will be controlled directly with a standard mouse/keyboard and can be launched on any standard Windows, Linux, or MacOS system. There are no special peripherals required to play.

4. System Features

4.1 Scoreboard

4.1.1 Description and Priority

A scoreboard to show the high scores of players on the machine, and/or from the internet. The scoreboard would also allow users to attach a 3-6 character string as their tag or name. this would be a nice personalization feature to be added after the bas game has been created. It is a medium priority feature.

4.1.2 Functional Requirements

The Scoreboard will utilize a text-based database to store the top 10 high scores for the system, as well as a 3-6 character tag or name.

4.2 Color Changing

4.1.1 Description and Priority

Grants the user the option to change the colors of Tetris pieces through either color selection, or through predefined "theme packs". The feature is of low importance and will be added following the implementation of the other features.

4.1.2 Functional Requirements

In order to implement this feature, it will require the creation of a theming screen and a modification to the settings view. There won't be any additional requirements associated with the feature.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

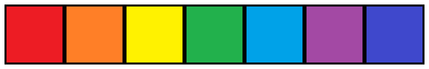
There should be no significant performance requirements for the game, it should be able to run on all systems that are capable of running Java.

5.2 Security Requirements

The regular and intended use of the game should not have any significant security issues. We cannot guarantee absolute security if the game is modified, or if the end-user's computer is already compromised.

5.3 Appendix A: Glossary

Active Piece: The Block on the game board that is falling, and the one the player actively controls.



Block/Piece/Tetromino: Used interchangeably to refer to the blocks that are used in play. There are 7 total pieces in the game. Each Block is composed of Squares.

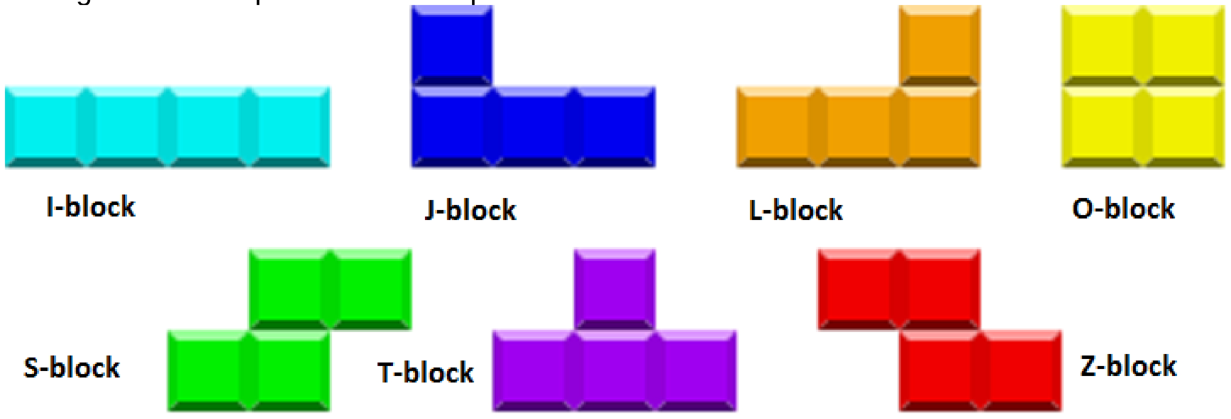
a. I : The I Block is comprised of 4 vertical Squares stacked in an I position.

b. L & J: These 2 Blocks are similar but mirrored. It is composed of 4 Squares (3 vertically stacked Squares with the bottom row having 2), in the shape of an L.

c. O: The O is Block composed of 2x2 Squares.

d. S & Z: These 2 Blocks are similar but mirrored. It is composed of 4 Squares in an S shape.

2. T: The T Block is composed of 4 Squares. 3 horizontal Squares with the middle Square having 2 vertical Squares in the shape of a T.



Matrix: Term coined by the original Tetris creators to refer to the playing field/game board. The game board is a 10x16 Square grid.

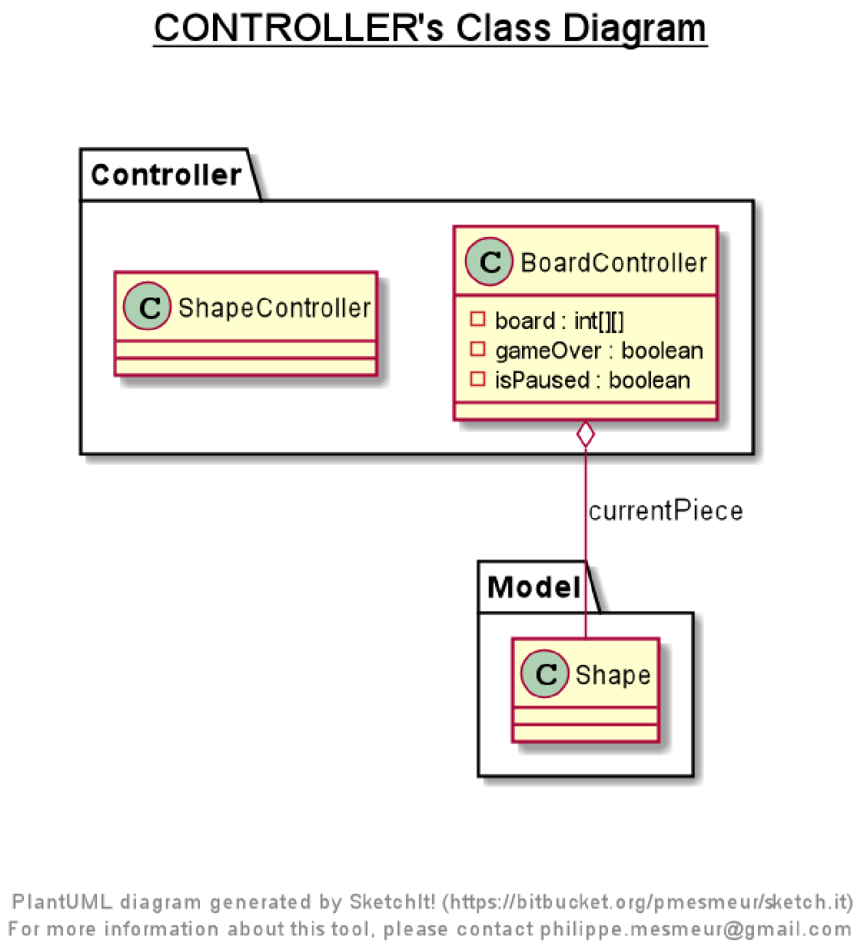
Main Menu: The screen where the player will be able to choose to either start a new game or change options.

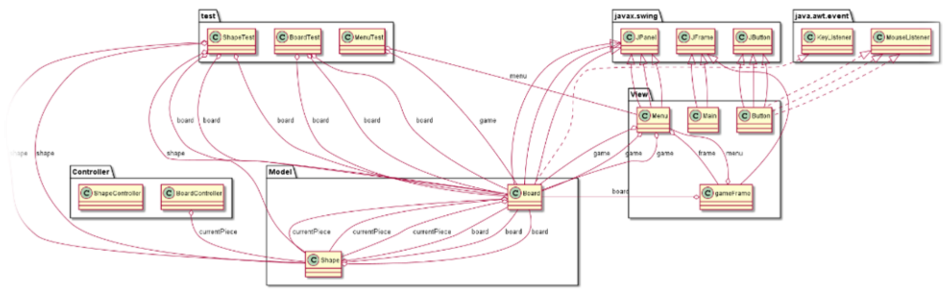
RNG: Random Number Generator; this is in reference to determine the next piece that will be put in play.

Rotate Button: The button used in gameplay to rotate the current piece in play. Henceforth, R1 corresponds to clockwise, and R2 to counterclockwise.

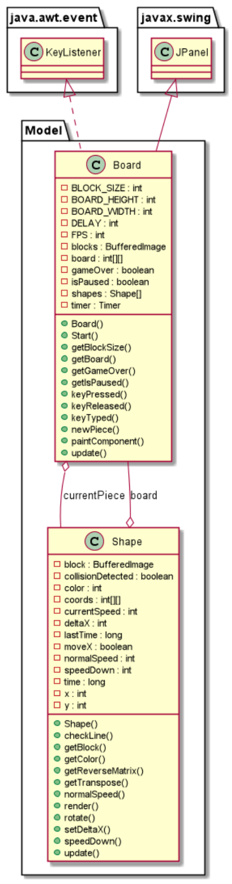
Score System: The system in the game that tracks the users score. An algorithm determines the amount of points the player gets per clear.

Square: The individual unit that each Block is comprised of.


CONTROLLER's Class Diagram
Controller
ShapeController
BoardController
board : int[][]
gameOver : boolean
isPaused : boolean
currentPiece
Model
Shape
PlantUML diagram generated by SketchIt! (https://bitbucket.org/pmesmeur/sketch.it)
For more information about this tool, please contact philippe.mesmeur@gmail.com

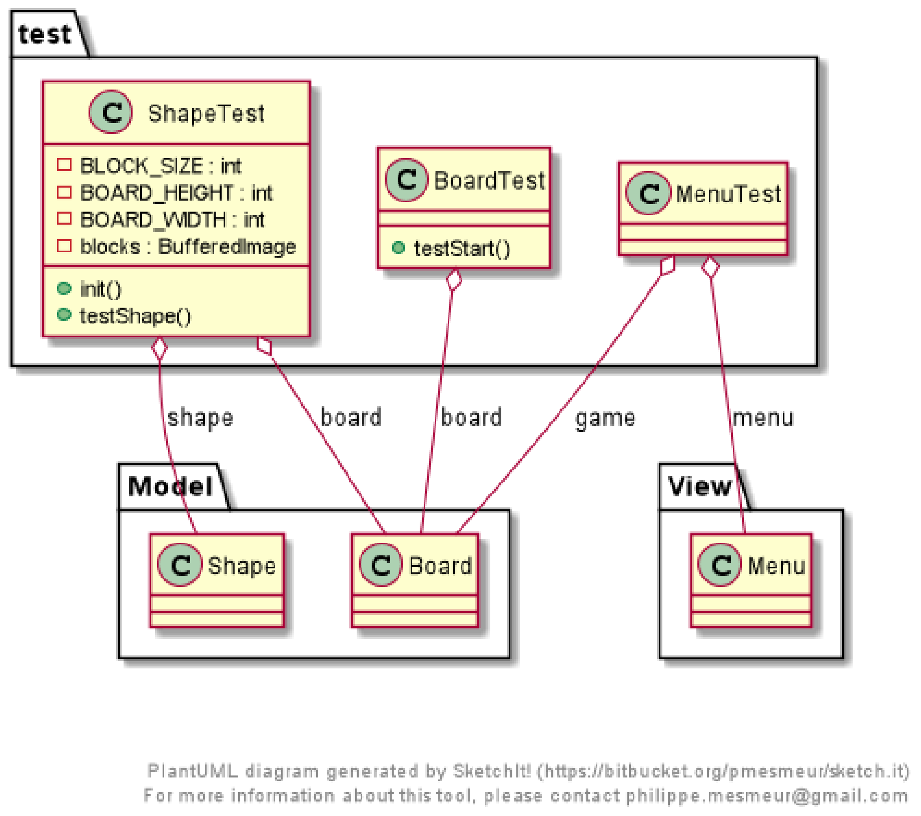


MODEL's Class Diagram

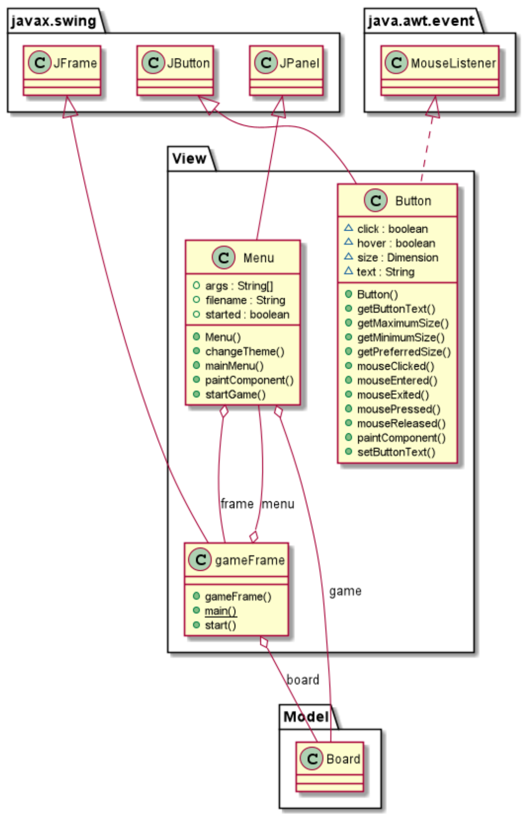


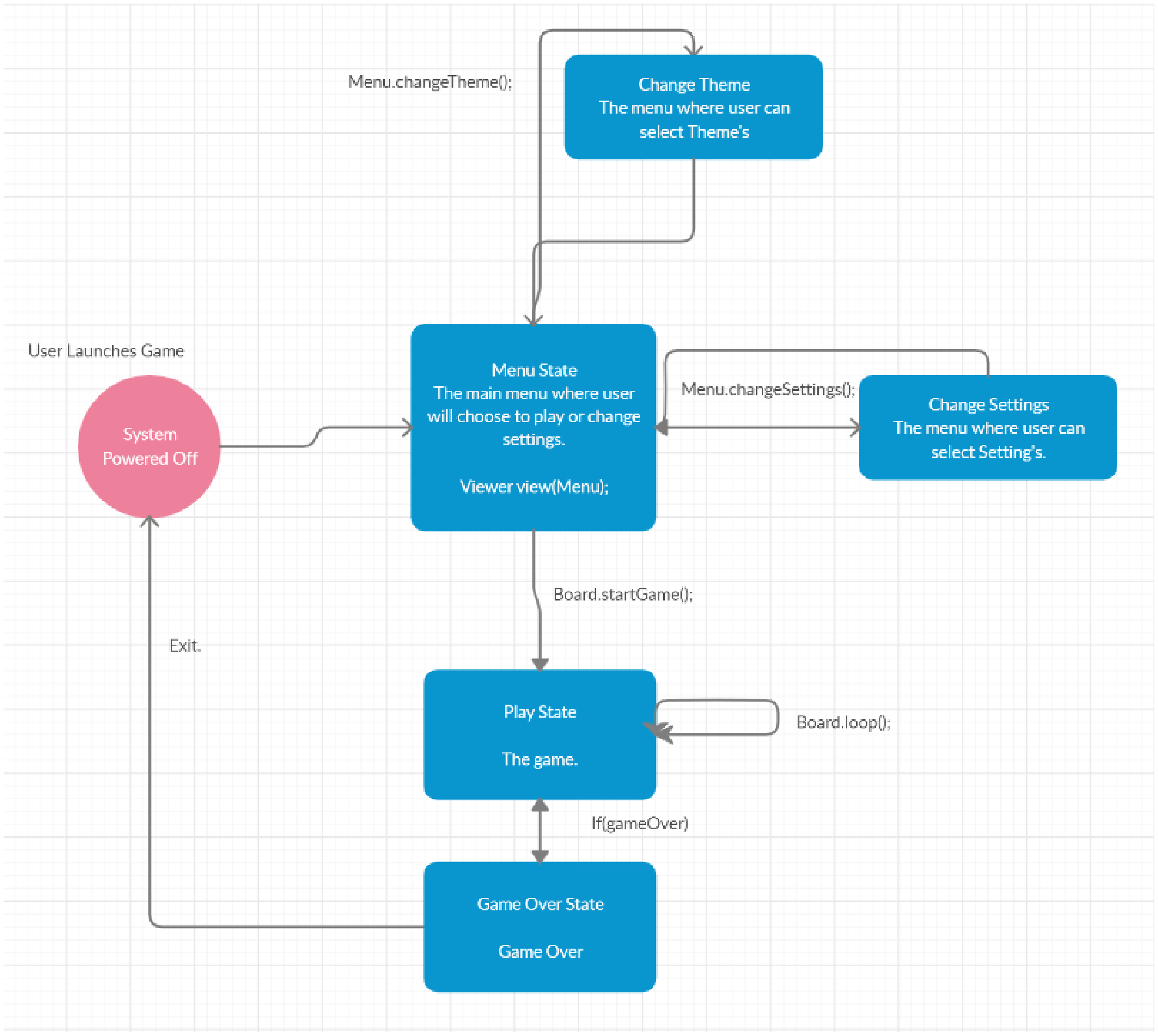
PlantUML diagram generated by SketchIt! (https://bitbucket.org/pmesmeur/sketch.it)
For more information about this tool, please contact philippe.mesmeur@gmail.com

TEST's Class Diagram



VIEW's Class Diagram





Appendix C: To Be Determined List

- Possible music application within gameplay as well as selection of user desired music.
- Tetris piece customization and introduction of new pieces.
- Profile implementation and player rank system.
- User interface upgrades through out the development of the a game.
- Additional game modes such as fast paced timed sessions or assisted gameplay.