SWEN30006 Software Modelling and Design
Project 1: Tetris Madness
- Project Specification -

School of Computing and Information Systems
University of Melbourne
Semester 2, 2022

Background

The classic arcade game Tetris has been around since the 1980's, and die-hard Tetris players are now seeking a new challenge. The game company Madness Industries™ has stepped in to fill this need with their new game design, ***Tetris Madness***. A previous team of Madness Industries has developed a replica of the simple Tetris game; their code works reasonably well with some documentation, but unfortunately it was built in a rush and was poorly designed. Madness Industries has recruited you and your team to revise the existing design and codebase and extend the simple Tetris game to include the new features that will make the game even madder.

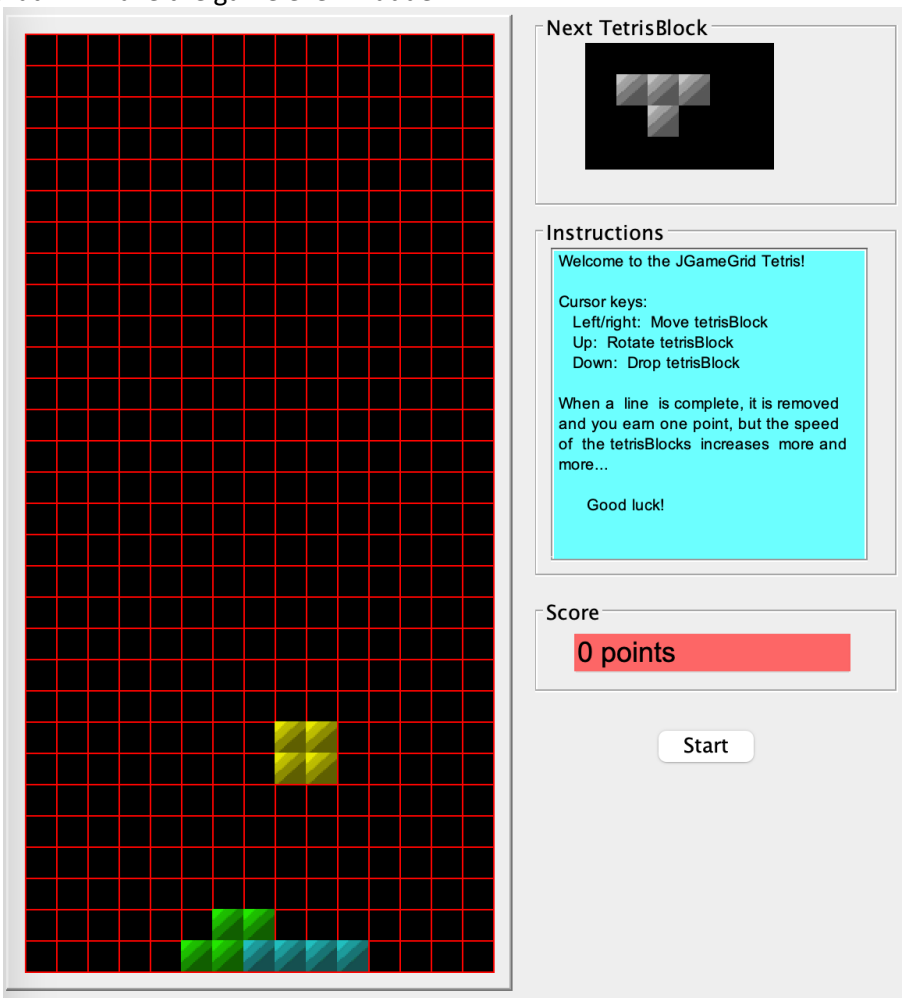

Figure 1: The GUI of the simple Tetris game

1. Use Cases

The use case text below describes the simple Tetris game that is already built and provided in the codebase (see 1.1), as well as Madness Industries' proposed extensions (see 1.2). The extensions have not yet been built and will be detailed further in the specification.

1.1 Playing Simple Tetris

Main Success Scenario:

1. The player opens the game
2. The game randomly selects a Tetris piece to be the current piece. There are 5 different shaped pieces (I, J, L, O, S T), each comprised of 4 individual blocks (called TetroBlocks).
3. The Tetris piece falls down the game grid at a particular speed.
4. As the piece is falling, the player controls the piece with keyboard actions
5. The piece continues to fall until it reaches the bottom of the game grid

Repeat steps 2-5

Alternate Scenarios

3a. If the player's current score surpasses set thresholds, the speed of the current piece increases.
4a. Left and right arrows to move the piece horizontally
4b. Up arrow to rotate the piece 90 degrees clockwise
4c. Down arrow to 'drop' the piece (increase its falling speed)
5a. If this results in grid row(s) being filled with TetroBlocks:
  5a1. The row(s) will be cleared
  5a2. The TetroBlocks above it will drop down
  5a3. The current score increases
  Repeat 5a1-5a3 for each row filled
5b. The game grid is vertically full such that a new piece cannot fall
  5b1. The game advises the player that the round is over
  5b2. The player selects to begin a new round
  5b3. The game grid is cleared and the score is reset to 0
*a. At any stage, the player exits the game

1.2 Extensions: Tetris Madness

Extension 1: Medium difficulty level

Extends Playing Simple Tetris
2a. When the difficulty is set to medium, the set of blocks will include 3 new shaped pieces (P, Q and Plus), each comprised of 5 TetroBlocks (see Figure 2)
3b. When the difficulty is set to medium, the speed will be 20% faster than the simple version (Note that the speed must still increase as the score increases)

Extension 2: Madness difficulty level

Extends Playing Simple Tetris
2a. When the difficulty is set to madness, the set of blocks will include 3 new shaped pieces, as per the Medium difficulty level
3b. When the difficulty is set to madness, each piece will fall at a random speed, within the range of [***S***, 2x ***S***], where ***S*** is the speed of the simple Tetris with respect to the current score.
4d. When the difficulty is set to madness, the rotate function will be disabled.

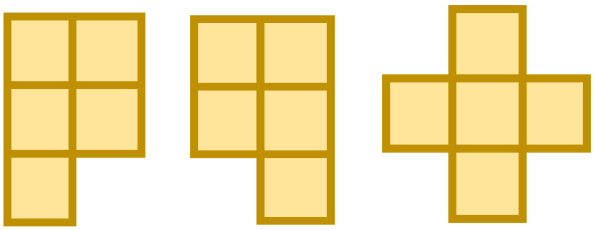

Figure 2: New shapes to be added as per Medium and Madness extensions

Extension 3: Recording statistics
Extends Playing Simple Tetris
5c. The game updates a statistics file that is kept about the game:
- 5c1. Update the counts of pieces played for the current round
- 5c2. Update the score for the current round
- 5c3. Update the average score per round

2. Your Task

1. Refactor the existing codebase
Madness Industries found that the simple Tetris game has a poor design as it is not currently very extensible. Part of your task is to refactor the existing code to meet software engineering design principles. A well-designed model will make the task of adding the new Madness features much simpler. Thankfully, the existing code is quite well documented with a design class diagram and code comments to assist you with understanding the code to start refactoring.

2. Add difficulty levels
As described in 1.2 (Extensions 1 & 2), the first new Madness feature is to make the game more challenging for Tetris players, with the addition of a **configurable** difficulty level. The difficulty level will be configured in the Properties file (an argument at a runtime). The defined levels will be:

Easy - The existing simple Tetris game, as per the Playing Simple Tetris use case

Medium - As per the Medium difficulty level extension

Madness - As per the Madness difficulty level extension

3. Add gameplay statistics
As described in 1.2 (Extension 3), Madness Industries would like to keep track of the gameplay to analyse the game plays. The system will be extended to write gameplay statistics to **a text file.**
Your output should be saved as '**Statistics.txt**' under the project root folder and can be overridden with each new game. The file should be formatted as follows:

```
Difficulty: Madness
Average score per round: 1234
----------------------------------------
Round #1
Score: 1234
I: 10
J: 20
K: 2
```

```
L: 0
O: 14
S: 14
T: 88
Z: 34
+: 0
P: 74
Q: 30
----------------------------------------
Round #2
Score: 1234
I: 10
J: 20
K: 2
L: 0
O: 14
S: 14
T: 88
Z: 34
+: 0
P: 74
Q: 30
```

3. The Base Package (Simple Tetris)

Getting started

- The package is provided as an IntelliJ project. You will need to unzip the archive and then 'open' the unzipped folder in IntelliJ. Note that the project requires Java 17.
- Then build and run the project. You will see the GUI appear and can play a game of simple Tetris using the standard keyboard actions.

System design

- The classes in the provided src package primarily handle the game behaviour, while the GUI operates by using the JGameGrid library.
    - The required changes for this project relate only to the application behaviour. You should not need to modify any code pertaining to the GUI, including the TetrisComponents class of the provided package.
- The TetrisGameCallback class logs game activity and is used for testing your system by the SWEN30006 staffs. Please ensure that any adjustments you make do not affect the output from this class.
- The system reads in a properties file that will specify the difficulty level, random seed, and auto-play configuration (for testing purposes). You can edit the properties in these files for your own testing, but make sure to preserve the default behaviour.
- A partial class diagram of the existing system has been provided to help understand the current design (see **Appendix 1**).

4. Project Deliverables

1 Extended version of the game (i.e., Tetris Madness): Submit your source code (with any/all libraries used included)

- Ensure that your code is well documented and includes your team name and team members in all changed or new source code files

2 Report: A Design Analysis report detailing the changes made to the project and the design analysis including identifying design alternatives and justifying your design decisions with respect to design principles. Specifically, your report should address the following points:

- (P1) An analysis of the current design: Explain the concerns of the current design and how it may hinder the extension of the new Madness features.
- (P2) Proposed new design of simple Tetris: Describe and justify how the simple Tetris game should be refactored to achieve a better design and/or facilitate future extensions.
- (P3) Proposed design of extended version (Tetris Madness): Describe and justify your design for the new Tetris Madness features that are added into your new game design.

3 Software Models: To facilitate the discussion of your report, the following software models should be provided in the report and used along with the discussions in P1-P3. You can use subdiagrams where appropriate.

- A domain class diagram for capturing the noteworthy concepts and their associations
- A design class diagram for documenting your ***new*** design for Tetris Madness
- A design sequence diagram for documenting the system's behaviour for one falling shape where the difficult level is configured as 'madness', no keyboard actions from a player, and statistic recording is enabled (i.e., as per 2 – 5, 2a, 3b, 4d and 5c described in 1.1)

5. Submission

All project deliverables should be included in the 1 project zip file with the specified structure (see Figure 3). **Only 1 member** in the team submits the file. Only the latest submission will be considered in case of multiple submissions from multiple people.

Testing Your Solution

We will be testing your application programmatically, so we need to be able to build and run your program without using an integrated development environment (IDE).[1] You must to ensure that the entry point must remain as **"src.Driver.main()".** You must not change the names of properties in the provided property file or require the presence of additional properties. If you add properties, they need to have defaults values as we will not set these in testing your submission. The test2 and test3 files also include a property to automatically move your Tetris pieces in a test mode. You need to ensure that any modification will not affect the original behaviour and result of these tests.

Note: It is your team's responsibility to ensure that you have thoroughly tested their software before submission.

[1] You do not need to be concerned about running without IDE if your system can be built and run using your IDE.

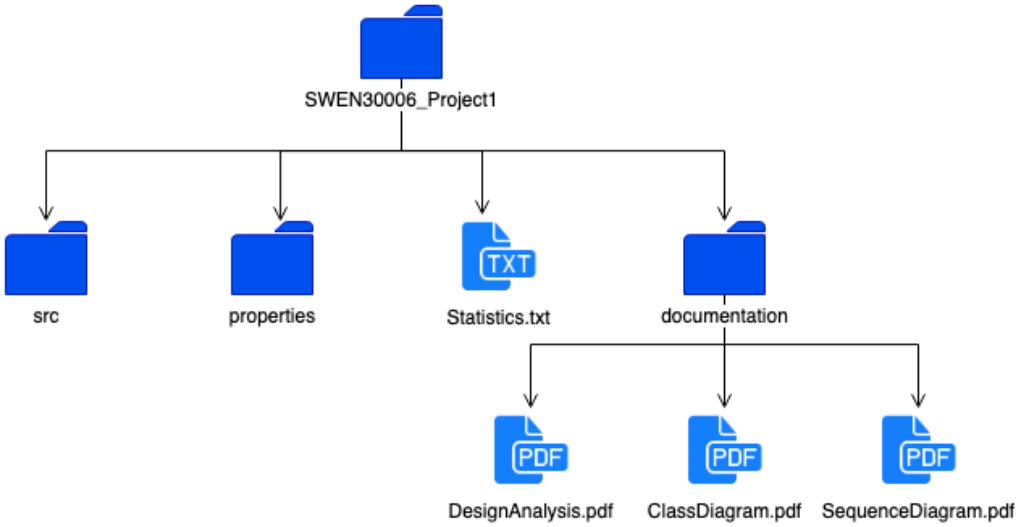

Figure 3: File structure for project submission

Appendix 1: Partial Class Diagram