TMGE Interface Document

Lancy Tan, Michael Wijangco, Samhitha Tarra,
Jules-Andrei Labador, Becky Dinh, Aliyah Clayton

University of California, Irvine
INF122: Software Design: Structure & Implementation
Dr. Jae Young Bang
March 14, 2022

About

The TMGE is a base code setup intended to simplify the process of developing a tile matching game, which is a game where users interact with tiles, typically with the intention of matching or removing them to win the game. This includes but is not limited to Tetris, Candy Crush, Columns, and others. As a developer, the TMGE can be used as a starting point to creating a game similar to these.

How To Use

The TMGE has superclasses with basic functionality encompassed by all tile matching games in each of the classes. For each of the following, the classes should be extended by a more specific subclass (if necessary) to allow for more specificity relative to the game being developed. Not every class or element in the TMGE is discussed in this document, and only the enhancements of the following superclasses is necessary for a new game implementation.

Tile

Use the Tile class to store information about each tile piece in the game. It already stored the X and Y coordinates for the tile as well as a generic type value, and it needs no modifications or extensions. Tiles should be stored in the Board class.

Board

The Board class should be extended from the superclass. Use the Board class methods to manage any data relating to the physical board of the game. In the TMGE it is represented as a 2D ArrayList of Tile elements. It can be used to create and modify board dimensions and check the bounds of the board. In the superclass, any unique board mechanics should be implemented here.

State

Use the State class to check if a win has been reached. This class should be extended for additional functionality regarding the win conditions of the specifics of the new game.

Game

This class needs to be extended for execution of the new game. This class presents all player-related options to the user, such as adding a new user, logging in, and viewing score. If additional player options are added, consider extending the Game class to add the option or change the user input menu. If not, simply modify the start() method to include the newly developed game as a player option, and override play() to start the play of the game in the subclass.

Main

Use this class to create a new instance of the game and call start() on it to run the game.