CS 371M

Homework 4: Tetronimoes

Submission:

All submissions should be done via git. Refer to the git setup, and submission documents for the correct procedure.

The root directory of your repository should contain your README file, and your Android Studio project directory.

Overview:

“If Tetris has taught me anything, it’s that errors pile up and accomplishments disappear.”
– Some guy on Reddit

In this and the next homework, you will be implementing a simple game of Tetris.

For those who have never played Tetris, definitely try it out. Below is a link to a free online tetris game. Use the “up” arrow key to rotate the blocks.

https://www.freetetris.org/game.php

On the following page is a screenshot of the game as it should look for this class.

For previous assignments, layout and quality of appearance have not been major factors in grading. However, appearance will be considered for this assignment. Your app should look at least as polished as the example app.

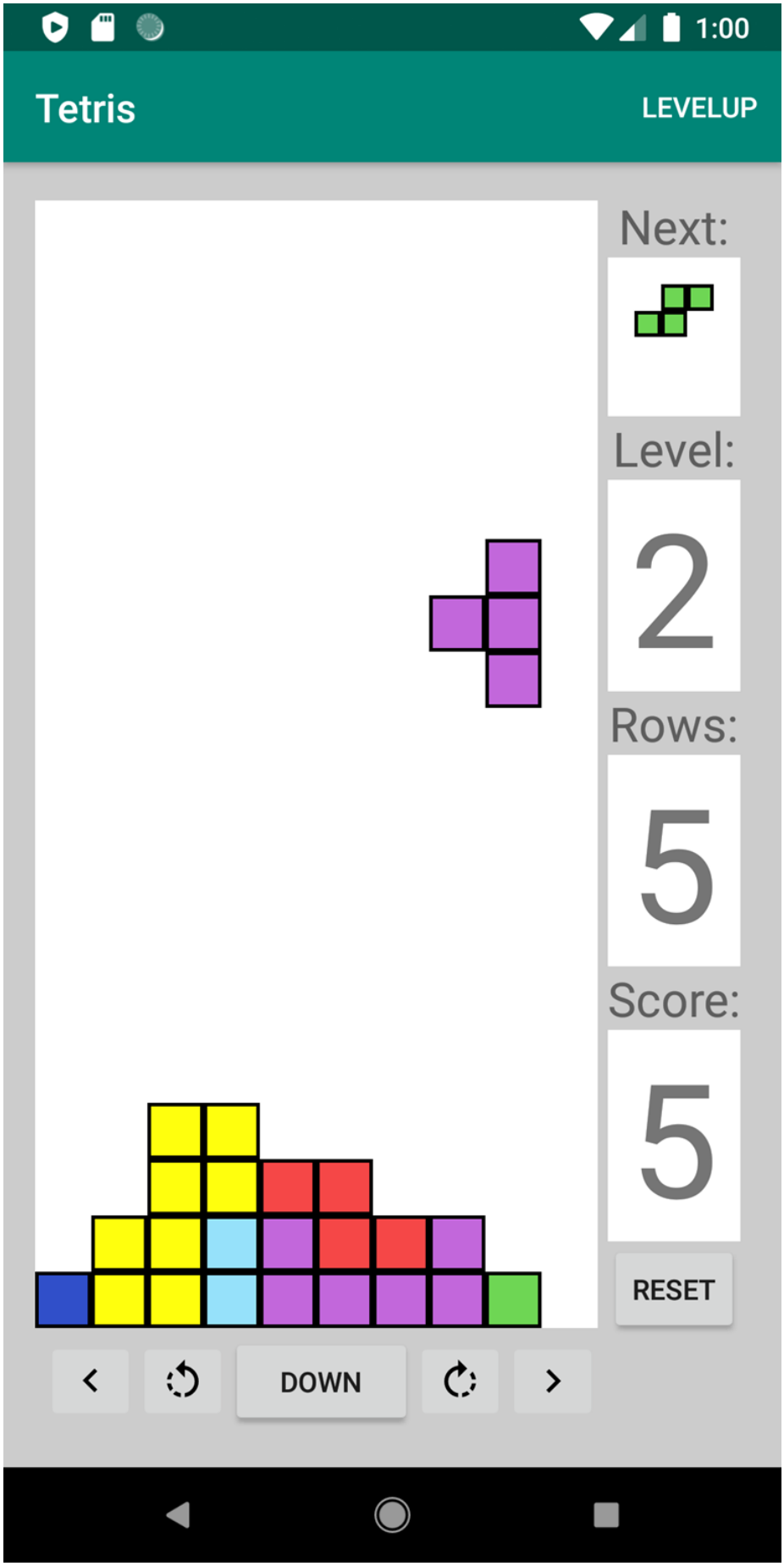
API:
We have defined several framework classes for your use.

TCell: A class that represents a singe square. Each TCell has a color and (x,y) coordinates.

TGrid: A class that creates a grid of cells. Use this to represent the main grid of cells in your game. This class also provides several utility functions that help you to detect and delete full rows of cells.

Tetromino: A class that represents a Tetris Tetromino. This class can be inserted into a TGrid, moved, and rotated. All Tetromino movement functions return a Boolean value depending on whether the movement succeeded or not.

TetrominoBuilder: This class returns a variety of fully constructed Tetrominoes.


1:00
Tetris
LEVELUP
Next:
Level:
2
Rows:
5
Score:
5
RESET
DOWN

Specifications:

Elements

Your app should meet the following requirements.

- It should have a grid 20 tall and 10 wide in which to play Tetris
  - There should be three invisible rows above the main grid. This is where you should place new Tetrominoes (so the full grid should be 23 tall).
  - Blocks should be square.
- Cells that are displayed on the screen should have a border.
- A window should display the next Tetromino that will appear.
- A text box with id **cleared** should display the current number of rows that have been cleared.
- A text box with id **score** should display the current score
  - Every time a row is cleared, add a number of points equal to the current level to the score. At level 1 add one point for each row, at level 2 two points, etc.
- A text box with id **level** that displays the current level.
  - Start at level 1.
  - Every time five rows have been cleared increase the level by one.
  - Every time the level is increased by one, decrease the time between each block movement by 20%
- A reset button with id **reset** (starts new game).
- An option with id **level_up** in the menu bar that allows the user to increase the level by 1.
- Control buttons: left with id **left**, right with id **right**, rotate left with id **rotate_left**, rotate right with id **rotate_right**, and down with id **down**.
  - It should be obvious to me by looking at your app which controls are which
  - left/right shift the tetromino to the left or right respectively
  - rotate should rotate the tetromino left or right.
  - down should cause the tetromino to drop immediately into place.
- We all know it's an Android app but keyboard control can be handy during testing. Use the following keys to control the tetromino: **a** for **left**, **s** for **down**, **d** for **right**, **z** for **rotate left** and **x** for **rotate right**. Hint: use key up event to detect key press.
- Display a toast: "You lose" when the game has ended.

Other Notes

At the beginning of your game, the time between each movement should be roughly one second. As the level increases this time will decrease.

When you insert new Tetrominoes, insert them into position (4,0). The next Tetromino is generated randomly with seed 3. To make testing more consistent, you can reset the random number generator when user clicks reset by calling TetrominoBuilder.resetRandom().

The game is over when you attempt to insert a Tetromino and cannot do so.

An instrumented unit test is provided in the androidTest dir. It reads in a command file stored in the assets directory and performs UI actions according to the commands in the file. This test is based on the Espresso Android UI testing framework. You are welcome to submit your own command file to show that your code is implemented correctly. The test hooks the UI elements by its id. To run the tests, you need to follow the naming convention or modify the tests. You can submit tests as part of your project. If you submit a test that is good enough to run on other student's assignments and if your tests find problems that ours do not, you will get extra credit.

Please fill in the provided README file.

Hints:

http://tetris.wikia.com/wiki/Tetris_Wiki

The coordinate system of a TGrid object starts in the upper left corner. The positive X axis points to the right, the positive Y axis points down.

Pay attention to the insertIntoGrid method in Tetromino. It inserts the Tetromino into a larger Tgrid(parent parameter). There're two larger TGrid in this game. One is the game board and the other one is the preview grid.

TGridView is the class that displays the "larger TGrid" for the Tetris game. You need to implement onDraw to draw the grid. When inserting tetromino into the larger grid, cells are copied from tetromino to the larger grid. If you draw the larger grid, it will draw all tetrominoes that have been inserted into the grid. The grid parameter initialized in TGridView is the larger grid.

The TGridView class creates and draws TGrid. When inserting a tetromino into the parent grid, the parent grid copies the cell from tetromino to itself. If you draw the parent grid, it will draw all tetrominoes that have been inserted into the grid.

Create a new class called TetrisBoard that inherits from both TGridView and from TimeAnimator.TimeListener. Instantiate this class from the XML (e.g., start an entry with <tetris.cs371m.tetris.TetrisBoard). Android will call the constructor for you.

Because you are instantiating the class from XML, you need to provide several constructors for TetrisBoard. But you write these constructors, and they can have default values that you control. Remember, you want 23 rows and 10 columns, but 3 of those rows are invisible.

TetrisBoard is now a convenient place to put a bunch of your game logic.

DemoDraw is a good example on how to draw a grid and change its position. For this assignment, it may be beneficial to draw your squares as a fraction of the view's width and height.

Once you figure out how to draw a cell, you can use that to implement CellVisitor defined in TGrid. TGrid's visitCells method will call the CellVisitor's visitCell on every cell, which draws the whole grid.

You can use TimeAnimator and implement the onTimeUpdate() method to update grids on game board. You can also use a Handler or coroutines, but I suggest avoiding those.

Call the .invalidate() function on a View to cause it to redraw itself.

Don't forget to check if multiple rows can be deleted.