Project: 2048 Game
Spring 2024

CS 61B

Ajay Sharma

Remark

If you would like to see the project source code or discuss implementation details, please contact me at ajay.sharma@berkeley.edu.

Overview

2048 is a single-player sliding-tile puzzle game played on a 4×4 grid, originally created by Gabriele Cirulli in 2014. Each turn, the player "tilts" the board in one of four directions, causing numbered tiles to slide as far as possible and merge when two tiles of equal value collide—doubling their value. The objective is to combine tiles strategically to reach the elusive 2048 tile while maximizing the cumulative score and keeping the board from filling up.

In this CS 61B project, I implemented the core game logic in Java. The rendering and input handling were provided; my responsibility was to realize the tile-movement, merge behavior, scoring, and move-validation rules in GameLogic.java, adhering to the project specifications and style guidelines.

Description

The game state is represented by a 4×4 two-dimensional array of integers, where 0 denotes an empty cell and positive powers of two represent tile values. On each valid move (tilt), tiles must:

- Slide all the way toward the tilt direction, collapsing gaps.
- Merge once per move when two tiles of the same value collide, producing a tile of double value and adding that amount to the score.
- Prevent multiple merges on the same tile in one move (no chain merges).
- Spawn a new tile (value 2 or 4) in a random empty cell if the board changed.

The project is organized into two packages: `game2048logic`, where you implement tilt and merge functions, score tracking, and move validation; and `game2048rendering`, which handles graphics and user input.

Methodology

- **Data Structures**: Use a 4×4 `int[][]` board. Zero values indicate empty spaces.
- **Tilt & Merge Logic**:
  1. Extract the affected row or column into a 1D array based on the tilt direction.
  2. Shift non-zero values toward the merge end, then scan adjacent pairs to merge equal values once.
  3. Re-compress to fill any introduced gaps and write back into the 2D array.
- **Scoring**: Maintain a running total in `GameLogic` that increments by the value of each merge.
- **Random Tile Generation**: After any successful move, invoke the provided spawner to place a new tile (2 with 90% probability, 4 with 10%) in a random empty cell.
- **Testing & Debugging**: Develop unit tests covering all four tilt directions, merge edge cases (e.g., multiple identical tiles), and no-move scenarios. Use the supplied test harness for automated validation.

Results and Insights

The completed implementation passes all autograder tests, correctly handling sliding, merging rules, and tile spawning. Manual playtesting confirms that scoring is accurate, moves are validated properly, and the game ends when no further moves are possible or when the 2048 tile is reached. This project reinforced the value of decomposing a complex behavior into modular array operations, writing thorough unit tests, and adhering to coding style guidelines in a collaborative codebase.