Project Report

Maryam Mohammadi

Kharazmi University

Fall - 1403

Algorithm

Tile Class

The Tile class represents a tile on the game board with a value and a state (_has_merged to track if the tile has merged in the current move).

Attributes:

- **_value**: The value of the tile (e.g., 1, 2, 4, 8, etc.).
- **_has_merged:** A flag to check if the tile has merged in the current move.

Methods:

1. **__init__(self, tile_value):**
   - **Input:** tile_value (the initial value of the tile).
   - **Output**: Initializes the tile object with a value and sets _has_merged to False.
2. **__repr__(self):**
   - **Output:** Returns a string representation of the tile.
3. **set_value(self, value):**
   - **Input:** value (new value for the tile).
   - **Output:** Sets the value of the tile.
4. **inc_value(self):**
   - **Output:** Increments the tile value and marks it as merged.
5. **has_merged(self):**
   - **Output:** Returns a boolean indicating if the tile has merged.
6. **reset_merged(self):**
   - **Output:** Resets the merge state of the tile to False.
7. **get_value(self):**
   - **Output:** Returns the value of the tile.
8. **get_tile_value(self):**
   - **Output:** Returns the tile value in the format 2 ** value (e.g., if _value = 3, returns 8).
9. **__str__(self):**
   - **Output:** Returns the tile value as a hexadecimal string representation.
10. **update(value):**

    Output: Updates the tile's visual properties.
11. **change_text(value):**

    Output: Changes the tile's text based on its value.

12. **change_fill(value):**

Output: Changes the tile's background color based on its value.

Board Class

The `Board` class represents the game grid (a 4x4 matrix) and contains methods for managing the board state, processing user moves, and handling tile placements/merging.

Attributes:

- grid: A 4x4 grid (list of lists) where each element is a Tile object or None.
- score: Tracks the current score of the game.
- merge_count: Tracks the number of merges that have occurred.

Methods:

1. **__init__(self, initial_state=None, initial_score=0, initial_merge_count=0)**:
   - **Input:** initial_state (optional 4x4 grid), initial_score (optional score), initial_merge_count (optional merge count).
   - **Output:** Initializes the board with the provided state or an empty board, sets the initial score and merge count.
2. **__repr__(self)**:
   - **Output:** Returns a string representation of the board's state, score, and merge count.
3. **__str__(self)**:
   - **Output:** Returns a string representing the full state of the board, including a user-friendly view of the board and metrics.
4. **add_random_tiles(self, n)**:
   - **Input:** n (number of random tiles to add).
   - **Output:** Adds n random tiles to empty positions on the grid, with a 90% chance of a tile being value 1 and a 10% chance of being value 2.
5. **make_move(self, move)**:
   - **Input:** move (a string representing the direction: 'UP', 'DOWN', 'LEFT', 'RIGHT').
   - **Output:** Executes the corresponding move on the board and returns True if a move was made.
6. **Movement functions (__go_up, __go_down, __go_left, __go_right)**:
   - **Output:** Handles the logic of moving tiles in the specified direction, including merging tiles with the same value.

7. **Tile scooting functions (__scooch_up, __scooch_left, __scooch_right, __scooch_down)**:
   - **Output:** Moves the tiles without merging them (i.e., makes the tiles move towards the available space).
8. **Tile merging functions (__go_up_1, __go_left_1, __go_right_1, __go_down_1)**:
   - **Output:** Handles the merging logic for tiles when two adjacent tiles have the same value.
9. **is_empty(self, x, y)**:
   - **Output:** Returns True if the tile at position (x, y) is empty.
10. **is_board_full(self)**:

- **Output:** Returns True if the board is full (no empty spaces).

11. **print_board(self)**:

- **Output:** Returns a user-friendly string representation of the board with tiles' values displayed.

12. **print_metrics(self)**:

- **Output:** Returns a summary string of the current score, merge count, and the highest tile on the board.

13. **reset_tile_merges(self)**:

- **Output:** Resets the merge state for all tiles.

14. **get_max_tile(self)**:

- **Output:** Returns the value of the highest tile on the board and its coordinates.

15. **export_state(self)**:

- **Output:** Exports the state of the board as a 2D list where each element is either None or the value of a tile.

Game Class:

- Responsible for the graphical user interface (GUI) using Pygame.
- Initializes the game screen and tiles.
- The update_tiles method updates the state of the tiles on the screen based on the board's grid.
- The draw_tiles method renders all the tiles on the screen.
- The convert_grid method transforms the internal grid into a format suitable for the graphical display.

Main Game Loop:

- The game loop runs continuously, waiting for user input (key presses) to move the tiles in various directions (up, down, left, right).
- Every move triggers tile merging or shifting, followed by the addition of a new tile.
- The game ends when no more valid moves are available, but this termination is not explicitly handled in the code.

Services Provided:

- **Tile Management:** The Tile class handles the creation and management of tile values, merging, and resetting.
- **Move Handling:** The Board class provides the core game mechanics for moving and merging tiles in all four directions.
- **Board Display:** The board is represented in both a compact format for metrics and a detailed view for user interaction.
- **Random Tile Addition:** The board can add random tiles at empty positions, simulating the randomness of the game.

Potential Future Additions:

- **Undo/Redo functionality:** To allow players to revert to previous board states.
- **Save/Load functionality:** Save and load game states to/from a file.
- **AI or Difficulty Levels:** Implement AI algorithms to challenge the player or varying difficulty levels.
- **Graphics/GUI:** Convert the board and tile display to a graphical user interface (GUI) for a better user experience.
- **Tile animations:** Add animations for tile movement and merging.