Minesweeper System Architecture Overview

EECS 581, Software Engineering II

Group 23: Navya Nittala, Kundana Dongala, Katie Nordberg, Vi Lara, Christina Sorensen

Introduction

This project implements the classic Minesweeper game in Python. The system allows players to interactively uncover cells, flag suspected mines, and attempt to clear the board without detonating any mines. This architecture is designed to clearly separate game logic (backend) from user interaction (frontend) to ensure modularity and extensibility

How to play

The goal of Minesweeper is to uncover all safe cells on the board without detonating any mines. Players interact with the board using mouse clicks:

- Starting the Game:
    - The board is initially hidden. Some cells contain mines, but their positions are unknown.
    - The first cell you click is guaranteed to be safe.
- Player Actions:
    - Left Click (Uncover a Cell):
        - Reveals the contents of the selected cell.
        - If the cell contains a mine → Game Over.
        - If the cell has 0 adjacent mines → neighboring cells automatically expand open.
        - Otherwise, it displays the number of adjacent mines.
    - Right Click (Flag a Cell):
        - Marks a cell as a suspected mine.
        - Right-click again to remove the flag. Flagged cells cannot be uncovered until unflagged.
    - Quit:
        - Close the game window to exit at any time.
- Winning the Game:
    - Successfully uncover all non-mine cells without triggering a mine.
    - Flags help track suspected mines and avoid accidental clicks.
- Tips:
    - Use numbers on uncovered cells to deduce nearby mine locations.
    - Place flags strategically to reduce risk when uncovering new cells

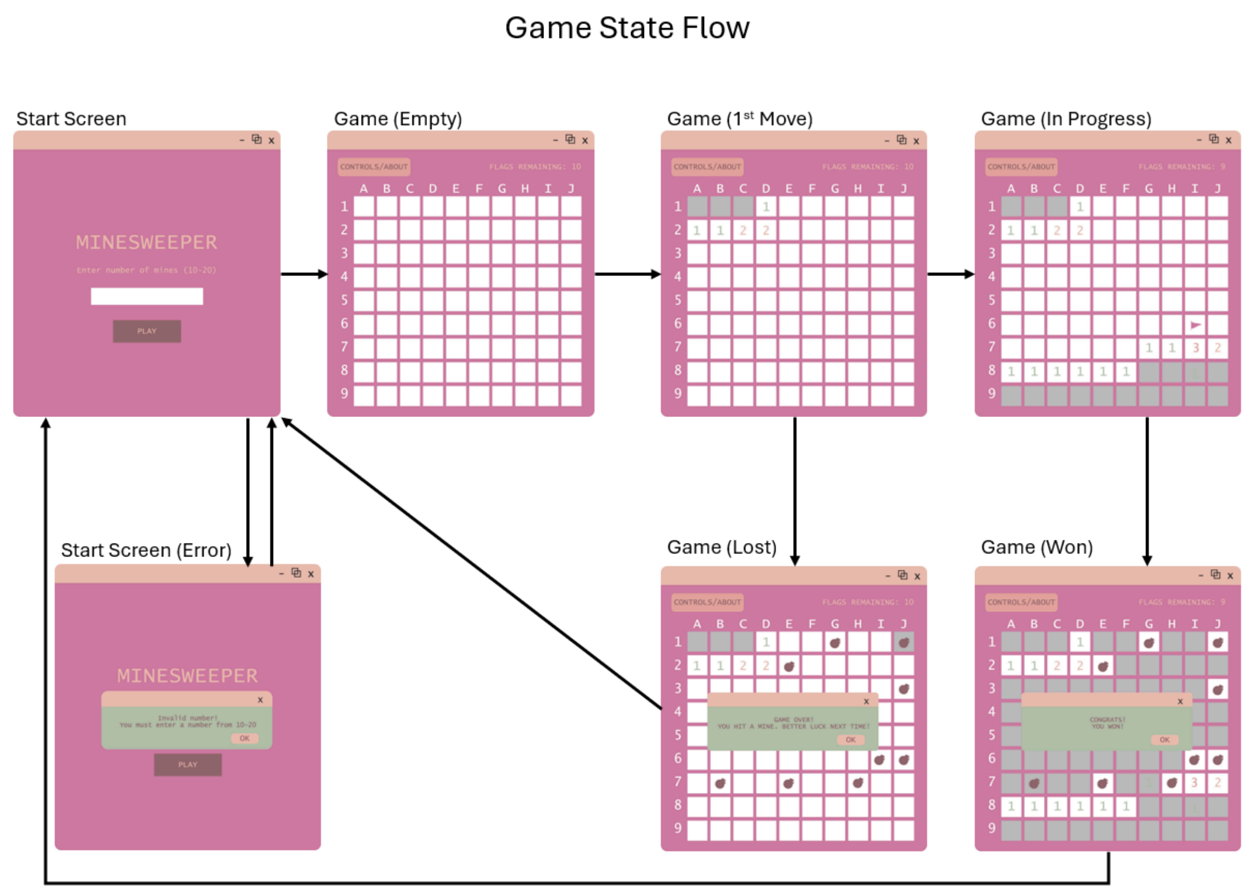

Figure 1: high-level diagram of the overall game state flow

High-Level Description

The Minesweeper game is implemented in Python with two main components:

- Game Logic (minesweeper.py):
    - Handles the core functionality of the

    - Minesweeper game, including board initialization, mine placement, cell uncovering, flagging, and game state management.
    - Implements the BoardManager class to encapsulate all board-related operations and state.
- Graphical User Interface (GUI) (minesweeper_tk.py):
    - Provides a user-friendly interface using Tkinter, Python's built-in GUI toolkit.
    - Tkinter was chosen because it is lightweight, easy to integrate with Python projects, requires no external dependencies, and provides enough flexibility to create an interactive grid-based board for Minesweeper.

        - It allows players to visually interact with the game by clicking cells to uncover or flag them, offering a more intuitive experience than console-based input.

Key Data Structures

- BoardContent:
    - A 2D array representing the Minesweeper board.
    - Each cell contains:
        - -1: A mine.
        - 0: No adjacent mines.
        - 1-8: Number of adjacent mines.
- BoardState:
    - A 2D array representing the visibility state of each cell.
    - Each cell contains:
        - 0: Covered.
        - 1: Flagged.
        - 2: Uncovered.
- Neighbors:
    - A list of relative positions used to calculate adjacent cells for mine placement and uncovering.

System Diagram

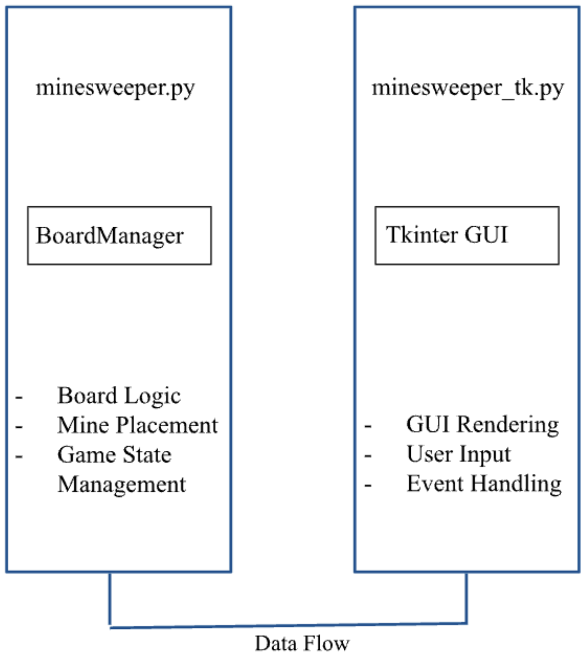

Figure 2: high-level diagram of the system components and data flow

Function Definitions

From minesweeper.py

BoardManager Class

- Purpose: Encapsulates all board-related operations and state.
- Attributes:
  - rows, cols, mines: Dimensions and number of mines.
  - boardContent: 2D array for mine and number placement.
  - boardState: 2D array for visibility state.
- Methods:
  - _initializeBoard:
    - Prepares the board for a new game by filling every cell in the grid with an initial value of zero, ensuring a clean starting state before mines are placed.
  - isMine:
    - Determines whether a given cell contains a mine by checking the boardContent grid and returning True if the cell is marked as a mine.
  - isCovered:
    - Checks if the player has not yet interacted with a specific cell by verifying that its status in the boardState grid is set to “covered.”
  - isFlagged:
    - Verifies whether a cell has been flagged by the player, typically as a suspected mine location, based on the boardState entry.
  - isUncovered:
    - Confirms that a cell has been revealed by the player, meaning its value in boardState indicates it is visible rather than hidden.
  - showBoardContents:
    - Displays the underlying logical structure of the board, including the positions of mines and the numeric counts of adjacent mines, primarily for debugging purposes.
  - showBoardState:
    - Outputs the current visible state of the board as the player would see it, showing which cells are covered, flagged, or uncovered.
  - flagCell:
    - Toggles the flag status of a cell, allowing the player to mark or unmark potential mine locations without uncovering them.
  - uncoverCell:

        - Reveals the contents of a cell; if it is the first move of the game, this method also triggers mine placement to guarantee the initial click is safe.
    - expandOpenCells:
        - Implements the flood-fill style expansion by recursively uncovering all neighboring cells when a zero-adjacent cell (no surrounding mines) is revealed, creating a chain reaction of safe openings.
    - placeMines:
        - Randomly distributes mines across the board after the player's first move, ensuring that the first chosen cell is not a mine, and then updates the numeric hints for all surrounding cells.

validatedIntInputInRange Function

- Purpose: Ensures user input is an integer within a valid range.
- Input: Minimum and maximum values, and a message.
- Output: Validated integer input.

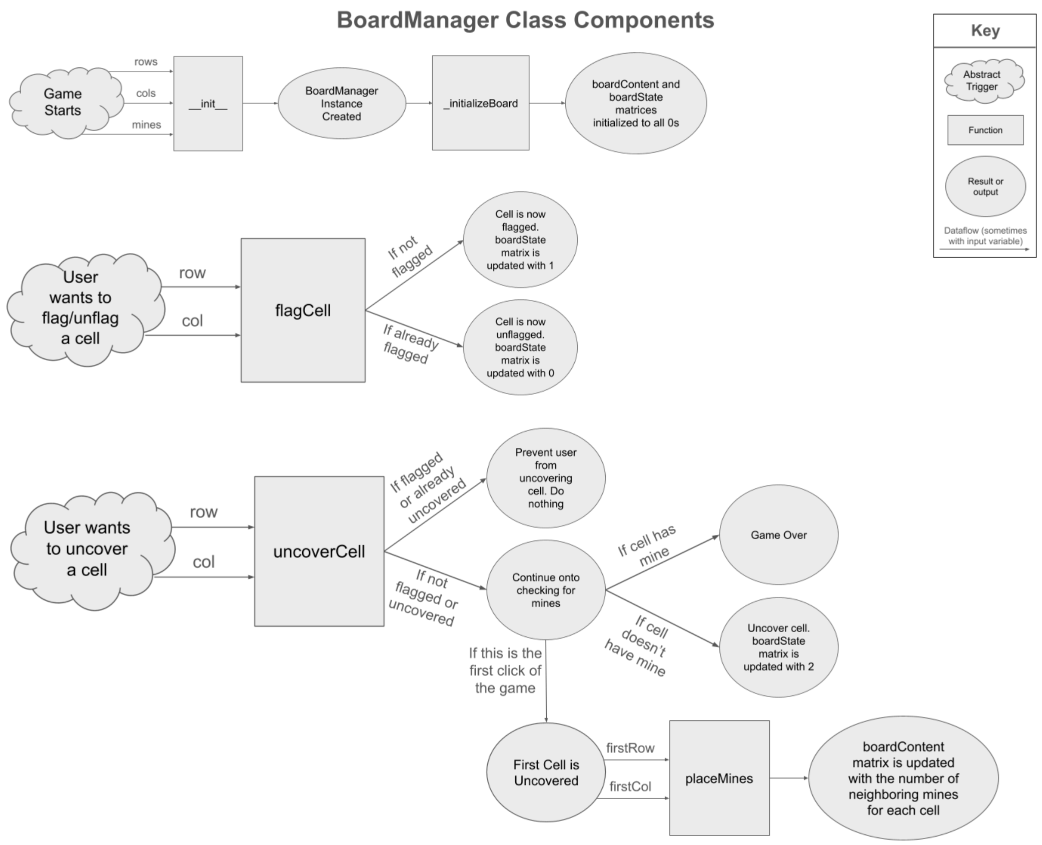
BoardManager Class Components
Key
Abstract Trigger
Function
Result or output
Dataflow (sometimes with input variable)
Game Starts
rows
cols
mines
__init__
BoardManager Instance Created
_initializeBoard
boardContent and boardState matrices initialized to all 0s
User wants to flag/unflag a cell
row
col
flagCell
If not flagged
Cell is now flagged. boardState matrix is updated with 1
If already flagged
Cell is now unflagged. boardState matrix is updated with 0
User wants to uncover a cell
row
col
uncoverCell
If flagged or already uncovered
Prevent user from uncovering cell. Do nothing
If not flagged or uncovered
Continue onto checking for mines
If cell has mine
Game Over
If cell doesn't have mine
Uncover cell. boardState matrix is updated with 2
If this is the first click of the game
First Cell is Uncovered
firstRow
firstCol
placeMines
boardContent matrix is updated with the number of neighboring mines for each cell

Figure 3: Detailed BoardManager Class Components Flow

From minesweeper_tk.py

Tkinter GUI Functions

- Purpose: Render the game board and handle user interactions.
- Key Functions:
  - create_board:
    - Builds and initializes the graphical board, creating the grid of buttons and setting up event bindings for user interaction.
  - on_cell_click:
    - Responds to user clicks on a cell, determining whether to uncover the cell or toggle a flag based on the mouse button used.
  - update_board:
    - Refreshes the graphical interface to reflect the latest board state, including uncovered numbers, flags, and revealed mines.
  - show_game_over:
    - Displays a message to the player indicating that the game is over, either due to winning or uncovering a mine

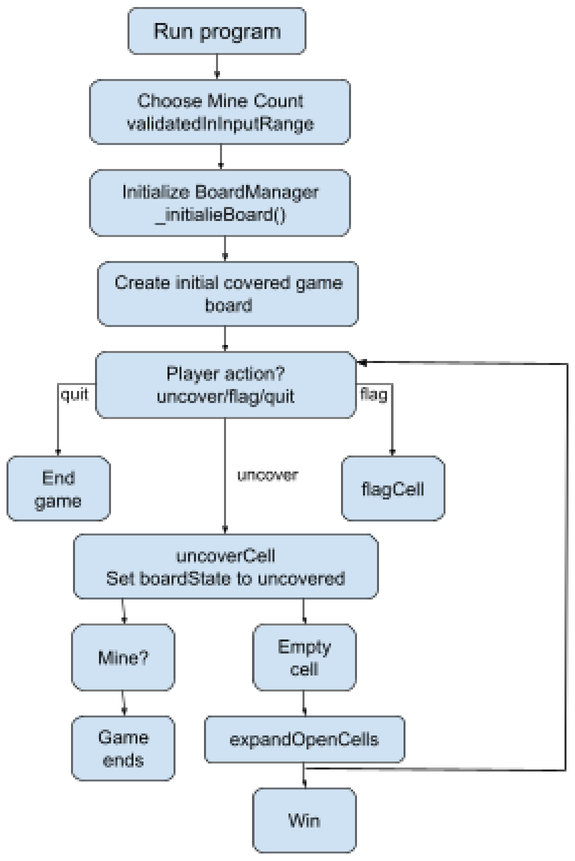

Figure 4: Detailed System Architecture design