Detailed Architecture

Used Libraries

- pygame
  - for UI and event handling
- pygame_gui
  - for the text box and the start button on the welcome screen

app.py

- Main game loop, loops while true
- handle events
  - wait for event to be QUIT(clicked the X in the top right to close window) then exit loop
  - handle game events, right-click, left-click, and ‘R'
    - there will be a special event on the welcome screen that we need to catch if it is the button we get from the draw_welcome() is clicked, this will be different than if left click because the button is created with pygame_gui
    - handle events by checking what type of event then calling the correct BoardGame function
      - for left-click you will need to call reveal()
      - for right-click you will need to call toggle_flag()
      - for 'R' you will need to initialize a new BoardGame instance
- Render the game screen
  - when we first render the screen or any time that we render the welcome screen we need to
  - receive the text box and button that the draw_welcome() will return
  - This will be a conditional based on the phase attribute in the BoardGame Class
    - Like if phase of BoardGame is ready display welcome screen

config.py

CELL_SIZE - in pixels

FPS - so that we dont get to 100% cpu usage

WINDOW_WIDTH, WINDOW_HEIGHT

GRID_ROWS, GRID_COLS

MINES_MIN, MINES_MAX

HELP_TEXT

COLOR_BG - the background color in (#, #, #) format

COLOR_CELL_COVERED, COLOR_CELL_UNCOVERED

COLOR_CELL_FLAG, COLOR_CELL_MINE, COLOR_GRID_LINES, COLOR_1_MINE, COLOR_2_MINE, COLOR_3_MINE(need COLOR_X_MINE up to 8)

FONT_NAME, FONT_SIZE

GRID_POS_X, GRID_POS_Y, HELP_TEXT_X, HELP_TEXT_Y, FLAGS_REMAIN_X, FLAGS_REMAIN_Y

board.py

- Data class Cell
  - is_revealed - boolean
  - is_mine - boolean
  - neighbor_mine_count - int
  - is_flag - boolean
  - can_be_mine - Boolean
    - This is for initializing so that we dont place a bomb where they first click
    - This should be defaulted to true, then later if needed flipped to false
- class BoardGame
  - attributes
    - total_mines
    - used_flags
    - board - 2d array of cells
    - is_first_click - boolean, set to true at beginning
    - phase - “ready” | “playing” | “won” | “lost”
  - functions
    - def init_board()
    - Randomly place bombs throughout the board
      - Before you place a bomb you will need to check if that cells can_be_mine value is true, if it is false you cannot place a mine there
    - def reveal(event)
      - This function will need to handle three main things

        - Is this the first click for this board
            - call the init_board() function
        - That the position of the click is valid, that is in the board
            - you can use the positions in the event object and check it against the boards position on the screen
        - If the click was valid then we need to handle it
            - is that spot already revealed, do nothing
            - is it a bomb, set phase to loss, call reveal all mines
            - is it a number(neighbor_mine_count >= 1), set that spots is_revealed to true
            - Is it an unrevealed empty spot, reveal all other empty cells next to this cell till you hit numbered cells
    - def check_win()
        - check if there are any empty spaces left that the user has not revealed
        - if they have won change the phase to “won”
    - def reveal_all_mines()
        - This will be called in the case that the player revealed and lost the game
            - simple loop to go flip the is_revealed value to true for all the mines
    - def toggle_flag()
        - places a flag on a cell, flags can only be placed
    - def handle_first_click(cell)
        - This function will take a cell as input
        - It will then set that cells can_be_mine to false as well as all its neighbors
        - It will set the is_first_click attribute of the board to false
        - Then it will call the initialize_board() function

view.py

- def draw_board(board)
    - takes in the game board, a 2D array and prints out the game board in the center of the screen
    - Will display the board by drawing rectangles with the rect() function from pygame
        - each rect will display differently based on the cell it is representing from the 2D array
            - states of the cell will have different colors based on state, colors will be defined in config.py

            - to display a number you will need to draw an empty background with the number ontop
    - Will also need to display:
        - how many flags
        - small section with descriptions of the controls of the game
- def draw_welcome()
    - this will create/draw the welcome screen
    - this will display:
        - Title of the game
        - welcome message
        - description of how to play the game
        - text box made with pygame_gui
            - this is for getting the number of bombs the users wants
        - A start game button also made with pygame_gui
        - This function will return both the text box and button