System Architecture Overview – Minesweeper

Program Summary

This project is a Python implementation of Minesweeper using the Pygame library. The system runs on a main event loop that listens for user input, updates the game state, and redraws the screen.

The program operates in four states:

- Menu – Displays title, buttons, and mine count selection.
- Playing – Core gameplay: reveal tiles, place flags, check win/lose.
- Win – Displays victory message, returns to menu on click.
- Lose – Displays loss message, returns to menu on click.

The architecture separates input handling, game logic, grid data, and rendering, while Pygame provides system-level services for drawing and event management.

Diagram 1 – High-Level Flow

Diagram 1 shows the overall data flow:

- User Input (mouse clicks, quit events).
- Event Handling interprets actions.
- Game State Manager determines whether the system is in Menu, Playing, Win, or Lose.
- Game Logic & Board System enforces rules (mine placement, recursive reveals, win/lose checks).
- Rendering Engine updates visuals (grid, buttons, counters, messages).
- User Feedback displays the new state on screen.

This diagram emphasizes the linear loop of input to processing to output.

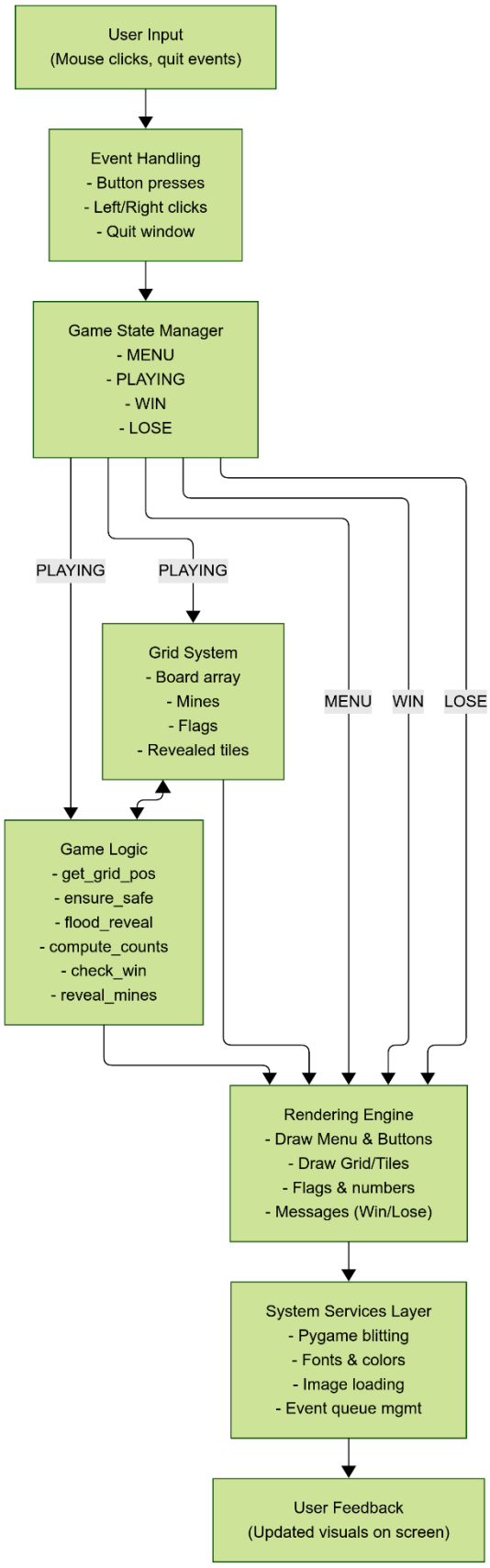
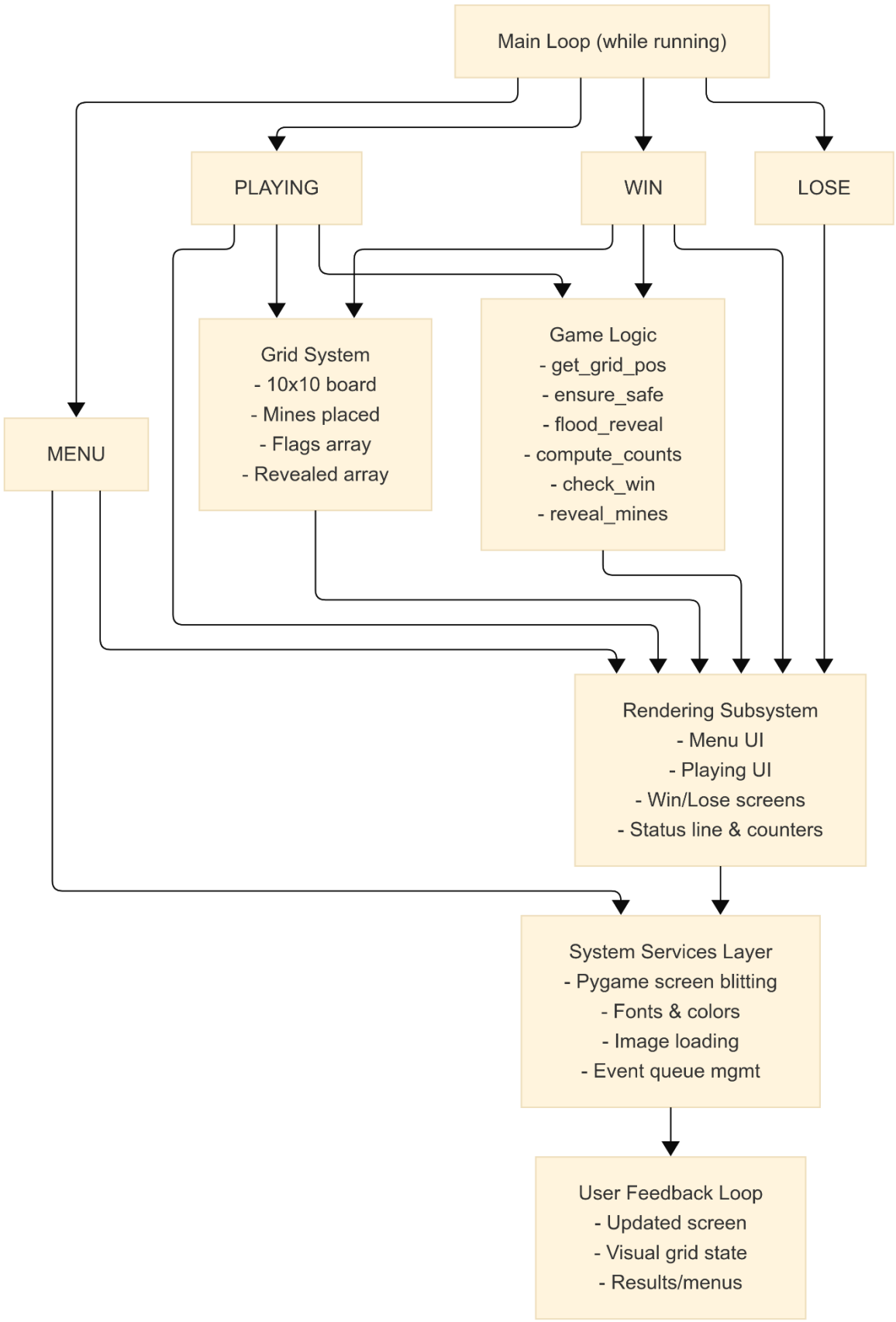

User Input
(Mouse clicks, quit events)
Event Handling
- Button presses
- Left/Right clicks
- Quit window
Game State Manager
- MENU
- PLAYING
- WIN
- LOSE
PLAYING
PLAYING
Grid System
- Board array
- Mines
- Flags
- Revealed tiles
MENU
WIN
LOSE
Game Logic
- get_grid_pos
- ensure_safe
- flood_reveal
- compute_counts
- check_win
- reveal_mines
Rendering Engine
- Draw Menu & Buttons
- Draw Grid/Tiles
- Flags & numbers
- Messages (Win/Lose)
System Services Layer
- Pygame blitting
- Fonts & colors
- Image loading
- Event queue mgmt
User Feedback
(Updated visuals on screen)

Diagram 2 – Component Interaction

Diagram 2 shows the internal interactions:

- The Main Loop drives all states (Menu, Playing, Win, Lose).
- In Playing, the Grid System (board arrays for mines, flags, revealed tiles) and Game Logic (rules and reveal logic) work together.
- All states feed into the Rendering Engine, which draws menus, grids, and messages.
- The System Services Layer (Pygame fonts, images, colors, event queue) supports rendering and input.
- The loop ends with User Feedback, showing updated visuals to the player.

This diagram emphasizes the interconnected nature of states, logic, and rendering.


Main Loop (while running)
PLAYING
WIN
LOSE
MENU
Grid System
- 10x10 board
- Mines placed
- Flags array
- Revealed array
Game Logic
- get_grid_pos
- ensure_safe
- flood_reveal
- compute_counts
- check_win
- reveal_mines
Rendering Subsystem
- Menu UI
- Playing UI
- Win/Lose screens
- Status line & counters
System Services Layer
- Pygame screen blitting
- Fonts & colors
- Image loading
- Event queue mgmt
User Feedback Loop
- Updated screen
- Visual grid state
- Results/menus