Process & Decision Documentation

Project/Assignment Decisions

Inspired by Minesweeper, I designed this project as a simplified tile-based puzzle game to directly align with the assignment's focus on using arrays and JSON data to generate and manage interactive game elements. The game board is constructed as a two-dimensional array of tile objects, with level parameters (such as grid size, tile size, padding, and mine count) defined externally in a JSON file. This separation allows the structure of the game to be data-driven rather than hard-coded.

By using arrays to represent the grid and JSON to control level configuration, the project emphasizes dynamic generation, iteration with loops, and structured data management. Game states such as Intro, Playing, Win, and Loss are used primarily to support interaction flow, while the core mechanics are governed by array traversalThis approach ensures that changes to level design can be made by modifying data rather than rewriting gameplay code.

GenAI Documentation

No GenAI used for this task.

Summary of Process (Human + Tool)

- Defined the game board as a two-dimensional array, with each element representing a tile object.
- Used JSON to store level configuration data (grid dimensions, tile size, padding, and mine count).
- Implemented loops to dynamically generate the grid and populate tiles based on JSON values.
- Created randomized mine placement by shuffling an array of grid positions and assigning mines accordingly.
- Used array traversal to handle tile interaction, win conditions, and mine revelation.
- Implemented simple UI states (Intro, Playing, Win, Loss) to manage when array-based interactions are enabled.
- Debugged issues related to input handling, state transitions, and array indexing.
- Iteratively refined visual layout and transparency to ensure the grid remains readable across all screens.

Decision Points & Trade-offs

- Chose to represent the board as a 2D array of objects rather than individual variables to simplify iteration and scalability.
- Used JSON-based configuration to control level parameters, trading immediate simplicity for better flexibility and clarity.
- Opted for randomized mine placement via array shuffling instead of fixed layouts to demonstrate array manipulation techniques.
- Limited tile logic to a binary outcome (safe or mine) to keep the focus on data structures rather than complex rule systems.
- Used minimal visual feedback so attention remains on how arrays and data drive game behavior.

Verification & Judgement

- Verified that the grid is correctly generated from JSON data by adjusting values and observing changes without modifying core logic.
- Tested array indexing and bounds checking to ensure tile selection maps accurately to mouse input.

- Confirmed that mine placement logic correctly assigns the expected number of mines using array operations.
- Checked that win and loss conditions correctly evaluate array state (revealed vs. unrevealed tiles).
- Evaluated whether the use of arrays and JSON clearly demonstrates structured data handling and dynamic generation.

Limitations, Dead Ends, or Open Questions

- The game uses a single level configuration at a time and does not yet support multiple JSON-defined levels.
- Tile data is reset between rounds rather than persisted, limiting long-term state complexity.
- The simplified mechanics do not fully explore more advanced array-based logic.
- Future iterations could expand JSON usage to support multiple levels, difficulty scaling, or different tile behaviors driven entirely by data.

Appendix

N/A