Process & Decision Documentation

Side Quests and A4 (Individual Work)

- Modified sketch.js and to generate a new level of checkerboard looking style using loops and array.
- Enhanced UI by changing instruction label color for clearer visibility.

Role-Based Process Evidence

🎯 **The Simple Answer:**

levels.json has 4 nested array levels:

1. **Root Object** `{ "levels": [...] }` - The container
2. **Array of Levels** `[Level1, Level2, Level3]` - Multiple mazes
3. **Array of Rows** `[Row0, Row1, Row2...]` - Horizontal strips in each maze
4. **Array of Tiles** `[1, 2, 0, 0, 1...]` - Individual squares in each row

📊 **Think of it like this:**

```
Building (JSON file)
  └── Floors (levels[0], levels[1], levels[2])
      └── Rows of Rooms (each array in a level)
          └── Individual Rooms (each number = one tile)
```

🔢 **What the Numbers Mean:**

- `0` = Floor (empty, walkable)
- `1` = Wall (solid obstacle)
- `2` = Start position
- `3` = Goal position

Visual Breakdown of Level 1

JSON Data:

```json
[
  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
  [1, 2, 0, 0, 0, 1, 0, 0, 0, 1],
  [1, 0, 1, 1, 0, 1, 0, 1, 0, 1],
  [1, 0, 1, 0, 0, 0, 0, 1, 0, 1],
  [1, 0, 1, 0, 1, 1, 0, 1, 0, 1],
  [1, 0, 0, 0, 0, 0, 0, 0, 3, 1],
  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
]
```

How It Looks as a Maze:

```
██████████    ← Row 0: All walls (top border)
█S···█···█    ← Row 1: Start (S) at column 1
█·██·█·█·█    ← Row 2: Interior walls
█·█·····█·█    ← Row 3: Pathways
█·█·██·█·█    ← Row 4: More obstacles
█·····G··█    ← Row 5: Goal (G) at column 8
██████████    ← Row 6: All walls (bottom border)

Legend:
█ = 1 (wall)
· = 0 (floor)
S = 2 (start)
G = 3 (goal)
```

```
41
42  function generateNewLevel(rows, cols) {
43    const level = [];
44
45    for (let r = 0; r < rows; r++) {
46      const row = [];
47
48  }
49
50  function preload() {
51    // loadJSON runs in preload so that by the time setup() runs,
52    // we can safely read levelsData.levels.
```

GenAI Documentation

Date Used: Feb 10, 2026

Tool Disclosure: Claude ai Sonnet 4.5

Purpose of Use: Help understand the structure of 'levels.json' file clearly and brainstorm maze layout.

Summary of Interaction: It helped me understand how the array of 'levels.json' file work before I created the new level and let the tool to suggest me simple maze layouts that uses array and loop.

Human Decision Point(s): I selected one maze layout out of the options GenAI suggested and rejected other options.

Integrity & Verification Note: I checked its accuracy through running the code and looking at the visual display.

Scope of GenAI Use: GenAI didn't making UI changes. To write the code, I used the suggestion (that uses 'tab' button) that is integrated in VSCode.

Limitations or Misfires: There was not limitations or misfires.

Appendix

My question:

> can you explain me how the levels.json file works? what does each array indicate?

GenAI's answer:

Great question! Let me break down exactly how the `levels.json` file works step-by-step.

Perfect! I've created a comprehensive guide explaining how `levels.json` works. Here's the key concept:

The Simple Answer:

levels.json has 4 nested array levels:

1. **Root Object** `{ "levels": [...] }` - The container
2. **Array of Levels** `[Level1, Level2, Level3]` - Multiple mazes
3. **Array of Rows** `[Row0, Row1, Row2...]` - Horizontal strips in each maze
4. **Array of Tiles** `[1, 2, 0, 0, 1...]` - Individual squares in each row

Think of it like this:

```
Building (JSON file)
└── Floors (levels[0], levels[1], levels[2])
    └── Rows of Rooms (each array in a level)
        └── Individual Rooms (each number = one tile)
```

What the Numbers Mean:

- `0` = Floor (empty, walkable)
- `1` = Wall (solid obstacle)
- `2` = Start position
- `3` = Goal position

Example:

javascript

```
levelsData.levels[0][1][1] = 2
                ↑  ↑ ↑ ↑
                │  │ │ └── Column 1
                │  │ └──── Row 1
                │  └────── Level 0 (first level)
                └────────── The levels array
```

This means: **In Level 1, at Row 1, Column 1, there's a Start tile (value 2)**

The guide includes visual diagrams, step-by-step breakdowns, and shows exactly how the code transforms these arrays into the maze you see on screen!

Understanding the Array Hierarchy

Level 1: Root Object

json

```
{ ← This is the ROOT OBJECT
  "levels": ...
}
```

What it is: A JSON object with one key called `"levels"`

Level 2: Array of Levels

json

```
"levels": [
  [ /* Level 1 grid */ ],
  [ /* Level 2 grid */ ],
  [ /* Level 3 grid */ ]
]
```

What it is: An array containing multiple level grids
Array Index:

- `levels[0]` = First level
- `levels[1]` = Second level
- `levels[2]` = Third level

What each element represents: A complete maze/level grid

Level 3: Array of Rows (One Level)

Let's look at Level 1:

json

```
[
  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1], ← Row 0 (top border)
  [1, 2, 0, 0, 0, 1, 0, 0, 0, 1], ← Row 1
  [1, 0, 1, 1, 0, 1, 0, 1, 0, 1], ← Row 2
  [1, 0, 1, 0, 0, 0, 0, 1, 0, 1], ← Row 3
  [1, 0, 1, 0, 1, 1, 0, 1, 0, 1], ← Row 4
  [1, 0, 0, 0, 0, 0, 0, 0, 3, 1], ← Row 5
  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]  ← Row 6 (bottom border)
]
```

What it is: An array of rows
Array Index:

- `level[0]` = First row (top of the maze)

- `level[1]` = Second row
- `level[6]` = Seventh row (bottom of the maze)

What each element represents: One horizontal row of tiles

Level 4: Array of Tiles (One Row)

Let's look at Row 1 of Level 1:

json

```
[1, 2, 0, 0, 0, 1, 0, 0, 0, 1]
↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
 0  1  2  3  4  5  6  7  8  9 ← Column numbers
```

What it is: An array of numbers representing individual tiles
Array Index:

- `row[0]` = First column (leftmost tile)
- `row[1]` = Second column
- `row[9]` = Tenth column (rightmost tile)

What each number means:

- `0` = Empty floor (walkable space)
- `1` = Wall (solid obstacle)
- `2` = Start position (where player begins)
- `3` = Goal position (where player needs to reach)

Visual Breakdown of Level 1

JSON Data:

json

```
[
  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
  [1, 2, 0, 0, 0, 1, 0, 0, 0, 1],
  [1, 0, 1, 1, 0, 1, 0, 1, 0, 1],
  [1, 0, 1, 0, 0, 0, 0, 1, 0, 1],
  [1, 0, 1, 0, 1, 1, 0, 1, 0, 1],
  [1, 0, 0, 0, 0, 0, 0, 0, 3, 1],
  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
]
```

How It Looks as a Maze:

■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ← Row 0: All walls (top border)
■ S · · · ■ · · · ■ ← Row 1: Start (S) at column 1
■ · ■ ■ · ■ · ■ · ■ ← Row 2: Interior walls
■ · ■ · · · · ■ · ■ ← Row 3: Pathways
■ · ■ · ■ ■ · ■ · ■ ← Row 4: More obstacles
■ · · · · · · · G ■ ← Row 5: Goal (G) at column 8
■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ← Row 6: All walls (bottom border)

Legend:
■ = 1 (wall)
· = 0 (floor)
S = 2 (start)
G = 3 (goal)

My question:

can you suggest me some simple maze layouts for the new level using loops and array?

GenAI's answer:

Simple Maze Layout Patterns for Level 3

Here are various maze layout ideas you can create using loops and arrays. Each pattern uses mathematical logic to place tiles systematically.

Pattern 1: Checkerboard / Grid Pattern

Visual Layout:

■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
■ S · · · · · · · · · ■
■ · ■ · ■ · ■ · ■ · ■ ■
■ · · · · · · · · · · ■
■ ■ · ■ · ■ · ■ · ■ · ■
■ · · · · · · · · · · ■
■ · ■ · ■ · ■ · ■ · ■ ■
■ · · · · · · · · · G ■
■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■

Description:

- Walls appear at regular intervals creating a grid
- Obstacles on even rows at even columns (or odd rows at odd columns)

- Creates a uniform, predictable pattern
- Easy to navigate but requires zigzag movement

Loop Logic Concept:

- Place wall when both row AND column are even/odd
- Creates evenly spaced vertical and horizontal corridors

Pattern 2: Vertical Stripes

Visual Layout:

```
■■■■■■■■■■■■■
■S·■··■··■·■
■··■··■··■·■
■··■··■··■·■
■··■··■··■·■
■··■··■··■·■
■··■··■··■·■
■··■··■··G■
■■■■■■■■■■■■■
```

Description:

- Vertical walls at regular column intervals
- Creates tall corridors running top to bottom
- Player moves between vertical channels
- Requires horizontal navigation to change lanes

Loop Logic Concept:

- Place wall when column number matches specific intervals (e.g., every 3rd column)
- All non-border rows have the same pattern

Pattern 3: Horizontal Stripes

Visual Layout:

```
■■■■■■■■■■■■■
■S·········■
■■■■·■■■■■■■
■···········■
■■■■·■■■■■■■
■···········■
```

Description:

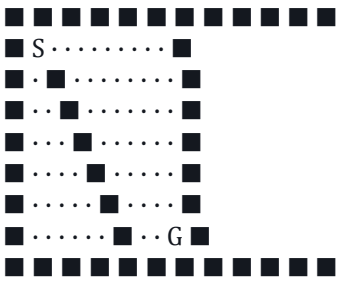
- Horizontal walls at regular row intervals with gaps
- Creates stacked horizontal chambers
- Player must find gaps to move between floors
- Forces vertical exploration

Loop Logic Concept:

- Place walls on specific rows, leaving gaps at certain columns
- Every few rows becomes a wall layer with doorways

Pattern 4: Diagonal Lines

Visual Layout:



Description:

- Walls form diagonal lines across the grid
- Creates angled barriers
- Visual interest with slanted obstacles
- Can create triangular open spaces

Loop Logic Concept:

- Place wall when row equals column (or row + column = constant)
- Creates 45-degree angle walls

Pattern 5: Concentric Boxes / Spiral

Visual Layout:

```
■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
■ S · · · · · · · · ■
■ · ■ ■ ■ ■ ■ ■ ■ · ■
■ · ■ · · · · · · ■ · ■
■ · ■ · ■ ■ ■ ■ · ■ · ■
■ · ■ · ■ · · ■ · ■ · ■
■ · ■ · ■ ■ · ■ · ■ · ■
■ · ■ · · · · ■ · ■ G ■
■ · ■ ■ ■ ■ ■ ■ · ■ · ■
■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
```

Description:

- Nested rectangular boxes moving inward
- Creates winding path from outside to center (or vice versa)
- Player spirals around to reach goal
- More complex navigation than linear patterns

Loop Logic Concept:

- Define layers (box 1, box 2, box 3...)
- Each layer is a rectangle with specific dimensions
- Leave gaps for entrance/exit to next layer

Pattern 6: Alternating Rows

Visual Layout:

```
■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
■ S · · · · · · · · ■
■ ■ ■ · ■ ■ ■ · ■ ■ ■ ■
■ · · · · · · · · · · ■
■ ■ ■ · ■ ■ ■ · ■ ■ ■ ■
■ · · · · · · · · · · ■
■ ■ ■ · ■ ■ ■ · ■ ■ ■ ■
■ · · · · · · · · · G ■
■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
```

Description:

- Every other row is mostly walls with gaps
- Open rows alternate with obstacle rows

- Creates horizontal layers with vertical passages
- Predictable but forces specific routes

Loop Logic Concept:

- On even-numbered rows, fill with walls except at gap columns
- Odd rows remain mostly open

Pattern 7: Cross / Plus Sign

Visual Layout:

```
■■■■■■■■■■■■■
■ S · · · · ■ · · · · ■
■ · · · · · ■ · · · · ■
■ · · · · · ■ · · · · ■
■■■■■■■■■■■■■
■ · · · · · · · · · · ■
■ · · · · · ■ · · · · ■
■ · · · · · ■ · · · G ■
■■■■■■■■■■■■■
```

Description:

- Vertical and horizontal walls intersect in middle
- Creates four quadrants
- Central crossing point is key navigation area
- Symmetrical design

Loop Logic Concept:

- Place wall on middle row (horizontal bar)
- Place wall on middle column (vertical bar)
- Leave pathways around the cross

Pattern 8: Border Maze (Thick Walls)

Visual Layout:

```
■■■■■■■■■■■■■
■■■■■■■■■■■■■
■■ S · · · · · · · ■■
■■ · · · · · · · · ■■
```

Description:

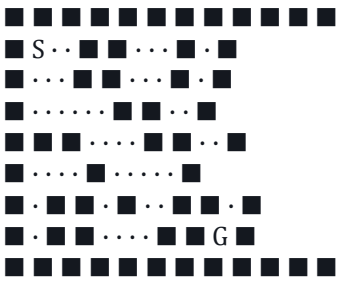
- Double-thick walls around perimeter
- Creates larger interior open space
- Could add interior obstacles
- Gives feeling of being in a walled courtyard

Loop Logic Concept:

- First two rows/columns are walls
- Last two rows/columns are walls
- Everything else is open (or has simple obstacles)

Pattern 9: Scattered Islands

Visual Layout:



Description:

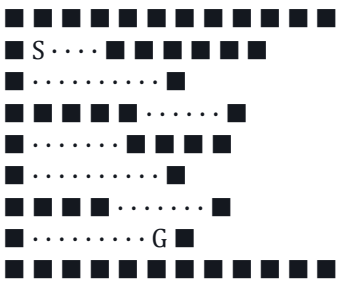
- Random-looking clusters of 2x2 wall blocks
- Creates "island" obstacles scattered throughout
- More organic, less geometric feel
- Multiple path options

Loop Logic Concept:

- Place 2x2 blocks at calculated positions
- Use math (like modulo) to create semi-random but reproducible placement
- Ensure blocks don't overlap or block all paths

Pattern 10: Zigzag Corridors

Visual Layout:



Description:

- Walls force S-shaped or Z-shaped path
- Creates winding corridors
- Longer path from start to goal
- Forces player to follow specific route

Loop Logic Concept:

- Walls shift position based on row number
- Creates stepped pattern moving left or right
- Each row's wall placement depends on previous row

Comparison Table

Pattern	Difficulty	Visual Style	Best For
Checkerboard	Easy	Geometric	Learning loops
Vertical Stripes	Easy	Linear	Column-based logic
Horizontal Stripes	Easy	Layered	Row-based logic
Diagonal Lines	Medium	Angular	Math relationships
Concentric Boxes	Medium	Nested	Complex loops
Alternating Rows	Easy	Rhythmic	Pattern recognition

Pattern	Difficulty	Visual Style	Best For
Cross/Plus	Easy	Symmetrical	Center-based logic
Border Maze	Very Easy	Open	Large playable area
Scattered Islands	Medium	Organic	Block placement
Zigzag Corridors	Medium	Flowing	Conditional logic

Recommendations

For Beginners:

- Start with Checkerboard, Vertical Stripes, or Border Maze
- These use simple modulo (%) operations
- Clear visual results help verify your code works

For Intermediate:

- Try Diagonal Lines, Alternating Rows, or Cross patterns
- Introduces conditional logic based on row/column relationships
- Teaches how to create symmetry

For Advanced:

- Concentric Boxes or Scattered Islands
- Requires nested loops and more complex positioning
- Creates more interesting gameplay

For Visual Impact:

- Zigzag Corridors or Spiral patterns
- Creates dynamic, flowing designs
- Looks more "hand-designed" than programmatically generated

Tips for Implementation

1. **Always start with borders:** Ensure row 0, row (max-1), column 0, and column (max-1) are walls
2. **Place start and goal first:** Set positions before adding obstacles

3. **Test your pattern:** Use small grids first (like 7x7) to see the pattern clearly
4. **Ensure paths exist:** Make sure player can actually reach the goal
5. **Use even/odd logic:** Row % 2 and column % 2 are powerful for creating patterns

---

Choose one pattern and I can help you implement it with actual loop code!