Side Quests and A4 (Individual Work)

GenAI Documentation

If GenAI was used (keep each response as brief as possible):

Date Used: Feb 9th, 2026

Tool Disclosure: ChatGPT 5.2 (OpenAI)

Purpose of Use:
I used GenAI as a coding assistant while I added a hazard tile mechanic and a third level to my maze game, with the goal of keeping my existing code intact.

Summary of Interaction:
I gave ChatGPT a strict prompt about what to add and what not to change, and it returned only the new pieces I needed. It suggested a hazard check method, a red hazard draw block, updated move-resolution logic, and guidance for the third level data format.

Human Decision Point(s):

I decided to represent the hazard with tile value 4 and to treat touching it as a fail state that sends the player back to level 1. I also chose to preserve my current controls, rendering style, and file structure, then designed the final level 3 layout and hazard placement myself.

Integrity & Verification Note:
I manually integrated all snippets into the correct files, checked syntax/placement, and verified gameplay behavior (goal advances level, hazard resets to level 1) before finalizing.

Scope of GenAI Use:
Used for code suggestion and implementation scaffolding only. File organization, integration, testing, debugging, and final acceptance were completed by me.

Limitations or Misfires:
Suggestions were generic and required manual adaptation to exact file context. Level design/difficulty and final data correctness still depended on human testing.

Summary of Process (Human + Tool)

I kept the original architecture and identified only the required changes: adding a hazard mechanic and a third level. I requested only net-new code blocks, inserted them manually into level.js and sketch.js, then added the third level data in levels.json using tile 4. After integrating everything, I tested movement outcomes and level transitions and iterated until behavior matched the specification.

Decision Points & Trade-offs

I wanted to avoid large edits, so I kept changes minimal to reduce regression risk. I used a tile-based hazard (4) instead of a separate obstacle system so the implementation stayed consistent with the existing setup. For fail behavior, I chose a full reset to level 1 because it creates a clear consequence, even though restarting only the current level would be less punishing. Visually, I reused the goal inset size and changed only the color so the hazard matched the game style.

Verification & Judgement

I made sure the movement and wall collision behavior stayed the same, and I verified that hazard checks run correctly before goal checks. I also confirmed that touching the red obstacle sends the player back to level 1, while reaching the goal still advances to the next level. Finally, I confirmed that the third level loads from JSON and renders correctly through the existing loop-based tile drawing.

Limitations, Dead Ends, or Open Questions

This implementation currently supports only one hazard behavior, so expanding to multiple obstacle types would require additional logic. The reset rule is fixed to level 1, and making that configurable could improve flexibility in future versions.

Appendix

My Prompt:

I am keeping the existing project architecture and reusing all current movement, collision, rendering loops, and level-loading flow. I only want to add a hazard mechanic and one new level. Please provide code that does exactly the following and nothing else:

1. Keep levels.json as the data source and add a third level so the game is 3 total levels.
2. Keep tile mapping 0=floor, 1=wall, 2=start, 3=goal, and add 4=hazard.
3. In Level (level.js), add isHazard(r, c) that returns true when tile value is 4.
4. In Level.draw(), render tile 4 as a red inset shape using the exact same size/position styling currently used for the goal tile (3).
5. In sketch.js input/move resolution, keep existing key handling and player.tryMove(...) logic. After a successful move, check tiles in this order:
    - if hazard (isHazard`) → loadLevel(0) (restart at level 1)
    - else if goal (isGoal) → nextLevel()

6. Do not refactor unrelated code, rename files, or change existing controls/visual style beyond the new red hazard tile and third-level data.

Return updated code blocks for level.js, sketch.js (only the changed section), and levels.json in my current format.

ChatGPT-5.2:

Great — here are the exact updates with no refactor and only the requested changes.

1) `level.js` — add this new method inside `class Level`

```
isHazard(r, c) {
  return this.tileAt(r, c) === 4;
}
```

---

2) `level.js` — add this new draw case inside your tile loop in `draw()`

```
if (v === 4) {
  noStroke();
  fill(220, 60, 60, 220);
  rect(
    c * this.ts + 4,
    r * this.ts + 4,
    this.ts - 8,
    this.ts - 8,
    6
  );
}
```

---

3) `sketch.js` — replace your current post-move goal check with this

```
if (moved) {
  const lvl = levels[li];

  // hazard first: restart at level 1
  if (lvl.isHazard(player.r, player.c)) {
    loadLevel(0);
    return;
  }
  // then normal goal progression
  if (lvl.isGoal(player.r, player.c)) {
    nextLevel();
    return;
  }
}
```

---

4) `levels.json` — add one new tile type in legend (comment/docs only)

4 = hazard (red obstacle)

---

5) `levels.json` — add a third level and include one 4 tile

Use your same array format and ensure:

- exactly one 2 (start)
- exactly one 3 (goal)
- at least one 4 (hazard)
- border walls still 1

If you want, paste just your new level array here and I’ll quickly validate it before you run.