Process & Decision Documentation

Project/Assignment Decisions

Side Quests and A4 (Individual Work)

Key Decision:

- I expanded Example 5 by adding multiple playable levels and non-playable transition screens using JSON data.
- A key decision was treating transition and end screens as additional “levels” inside levels.json, rather than hardcoding game states.

Role-Based Process Evidence

Name: Jenny South

Role(s): Designer + Developer (Individual Side Quest)

Implementing data-driven level progression, including goals, difficulty progression, and transition screens.

Goal of Work Session

I wanted to prove that I could:

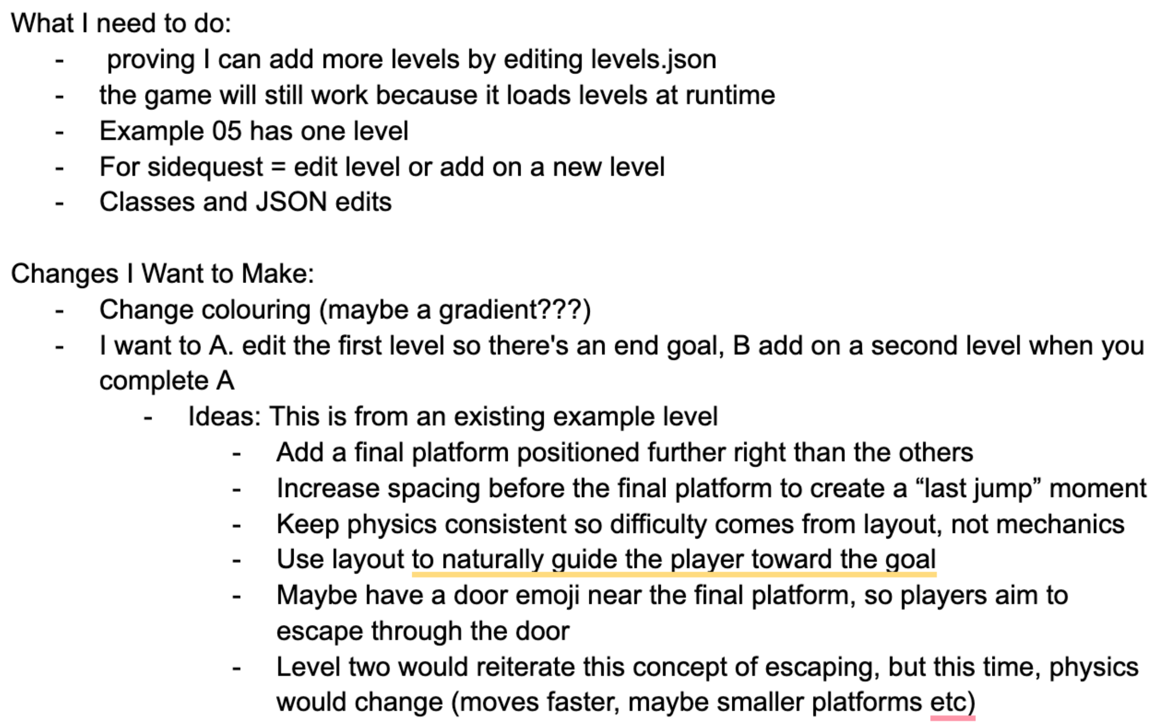
- Add and modify levels by editing JSON
- Increase difficulty through layout and physics rather than new mechanics
- Implement clear start, transition, and end states
- Maintain correct level progression without breaking existing gameplay logic

Tools, Resources, or Inputs Used

- p5.js (movement/animation/interaction)
- Visual Studio Code
- ChatGPT Version 5.2
- Example 5 starter code (week 04)

Process Evidence:

Screenshot 1: Wrote out my ideas on a Google Doc to brainstorm


What I need to do:

- proving I can add more levels by editing levels.json
- the game will still work because it loads levels at runtime
- Example 05 has one level
- For sidequest = edit level or add on a new level
- Classes and JSON edits

Changes I Want to Make:

- Change colouring (maybe a gradient???)
- I want to A. edit the first level so there's an end goal, B add on a second level when you complete A
    - Ideas: This is from an existing example level
        - Add a final platform positioned further right than the others
        - Increase spacing before the final platform to create a "last jump" moment
        - Keep physics consistent so difficulty comes from layout, not mechanics
        - Use layout to naturally guide the player toward the goal
        - Maybe have a door emoji near the final platform, so players aim to escape through the door
        - Level two would reiterate this concept of escaping, but this time, physics would change (moves faster, maybe smaller platforms etc)

This then led to my inputting these ideas and asking for a prompt I could use to best communicate my ideas.

My Prompt: You are helping me implement a sidequest in my existing Example 05 p5.js project. IMPORTANT: Do NOT invent any new level ideas or propose creative layouts. Only explain the coding process and where to edit files to achieve the exact changes I describe.

Context:

- My project loads levels at runtime from levels.json using loadJSON.
- Example 05 currently has 1 level in levels.json.
- The world is generated dynamically from JSON via classes (WorldLevel / Platform / BlobPlayer or similar).

What I need to prove:

- I can add/edit levels by editing levels.json only (data-driven).
- The game still works because it loads levels at runtime and can switch to the next level.

Changes I want to implement (explain step-by-step):

A) Add an end goal to Level 1:

- Define a goal object in the Level 1 JSON (e.g., goal position and size).

- Draw a visible goal marker (door emoji is okay) at that goal location.
- Detect when the player reaches the goal (collision or distance threshold).
- When the goal is reached, trigger “level complete”.

B) Auto-load Level 2 when Level 1 is complete:

- Add a second level object to levels.json (same schema as level 1).
- Update the code so “level complete” advances currentLevelIndex and reloads the next level from the already-loaded JSON data.
- Reset player + world state cleanly on level switch.
- Keep the manual ‘N’ key level advance if it already exists, but also support automatic advancing on goal completion.

Visual styling change:

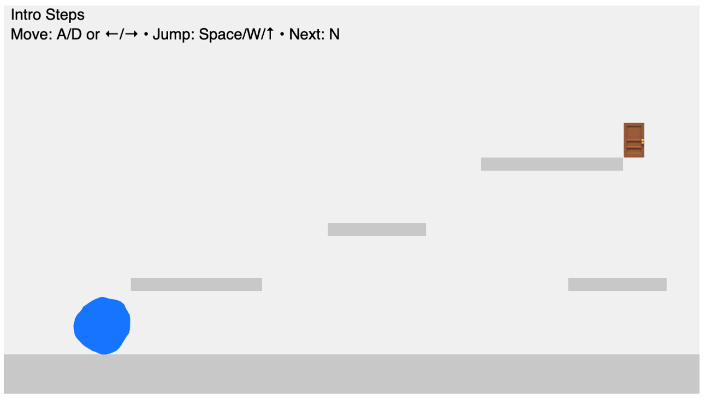
- Explain how to change the level theme colours from JSON (existing theme fields).
- If gradient background is not supported already, explain the simplest way to implement a gradient background in draw() or in the level rendering code, and how to keep it tied to level JSON (e.g., theme.bg1 and theme.bg2). Do not design the colours for me, just explain what fields to add and how to read them.

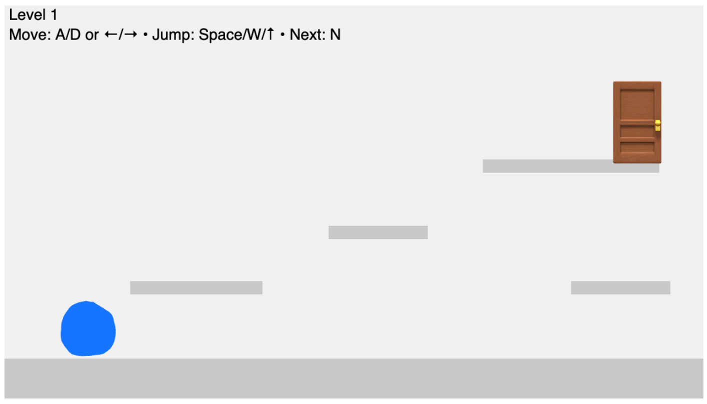
Constraints:

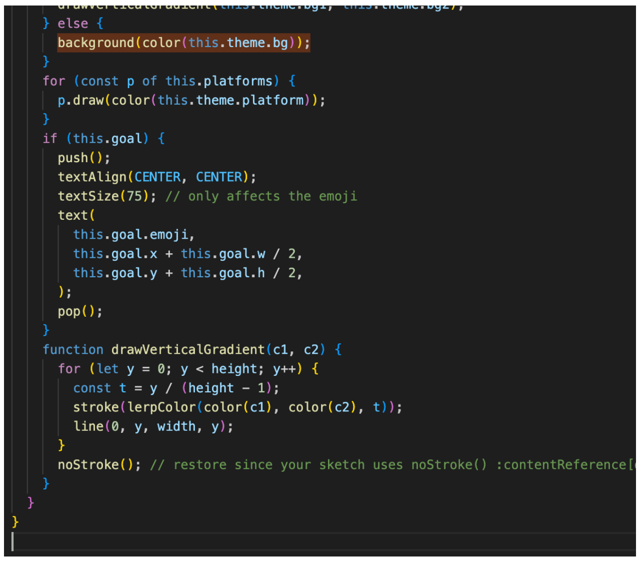
- Use my existing file structure (sketch.js, worldLevel.js, platform.js, blobPlayer.js, levels.json).
- Tell me exactly which file and which function to edit (e.g., “in sketch.js inside draw()…”).
- Provide small focused code snippets only when necessary (no full rewrites).
- Don’t change unrelated gameplay.
- Assume p5.js.

Screenshot 2: JSON iteration and level layout changes

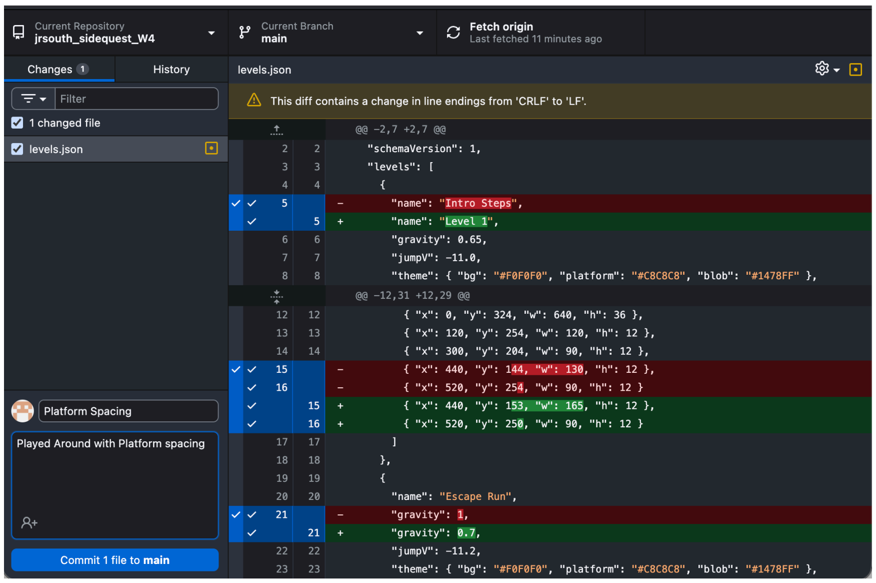
- Made many changes to the layout of platforms, spacing and physics of the blob, along with sizing and placement of the door


Intro Steps
Move: A/D or ←/→ • Jump: Space/W/↑ • Next: N


Level 1
Move: A/D or ←/→ • Jump: Space/W/↑ • Next: N



```
      drawVerticalGradient(this.theme.bg1, this.theme.bg2);
    } else {
      background(color(this.theme.bg));
    }
    for (const p of this.platforms) {
      p.draw(color(this.theme.platform));
    }
    if (this.goal) {
      push();
      textAlign(CENTER, CENTER);
      textSize(75); // only affects the emoji
      text(
        this.goal.emoji,
        this.goal.x + this.goal.w / 2,
        this.goal.y + this.goal.h / 2,
      );
      pop();
    }
    function drawVerticalGradient(c1, c2) {
      for (let y = 0; y < height; y++) {
        const t = y / (height - 1);
        stroke(lerpColor(color(c1), color(c2), t));
        line(0, y, width, y);
      }
      noStroke(); // restore since your sketch uses noStroke() :contentReference[
    }
  }
}
```



Played Around with Platform spacing

levels.json

This diff contains a change in line endings from 'CRLF' to 'LF'.

```
@@ -2,7 +2,7 @@
   "schemaVersion": 1,
   "levels": [
     {
-      "name": "Intro Steps",
+      "name": "Level 1",
       "gravity": 0.65,
       "jumpV": -11.0,
       "theme": { "bg": "#F0F0F0", "platform": "#C8C8C8", "blob": "#1478FF" },
@@ -12,31 +12,29 @@
         { "x": 0, "y": 324, "w": 640, "h": 36 },
         { "x": 120, "y": 254, "w": 120, "h": 12 },
         { "x": 300, "y": 204, "w": 90, "h": 12 },
-        { "x": 440, "y": 144, "w": 130, "h": 12 },
-        { "x": 520, "y": 254, "w": 90, "h": 12 }
+        { "x": 440, "y": 153, "w": 165, "h": 12 },
+        { "x": 520, "y": 250, "w": 90, "h": 12 }
       ]
     },
     {
       "name": "Escape Run",
-      "gravity": 1,
+      "gravity": 0.7,
       "jumpV": -11.2,
       "theme": { "bg": "#F0F0F0", "platform": "#C8C8C8", "blob": "#1478FF" },
```

GenAI Documentation

Date Used: Feb 04, 2026

Tool Disclosure: ChatGPT 5.2 (OpenAI)

Purpose of Use: To understand how to structure level progression, implement transition screens, and debug issues related to level completion and sequencing.

Summary of Interaction: The level ideas, progression, and screen integration came from my own brainstorming. I used GenAI to explain the coding steps instead of creating new mechanics or level designs for myself. It helped me understand how to set up transition and end screens as extra “levels” using a type field in JSON, and how to use a next index to control level order. It also helped me figure out why levels were skipping or completing right away by explaining the draw loop timing and goal detection logic.

Human Decision Point(s): While GenAI helped explain the code and support debugging, all creative and structural decisions were made by me. I determined the level layouts, how difficulty increased between levels, and how the transition screens concept was visually communicated. After understanding the source of the issues, I also decided which debugging strategies to apply, such as adding a short delay before goal detection.

Integrity & Verification Note: All changes discussed with GenAI were manually implemented and tested in p5.js. I checked that levels loaded in the correct order, that gameplay paused on transition and end screens, and that pressing SPACE moved to the correct next level. I also played through the game multiple times to confirm that goals no longer triggered immediately and that level progression worked as expected.

Scope of GenAI Use: GenAI was used to help me conceptualize my ideas of levels, transitions, screens and debugging. I would provide it with my detailed prompts so it would be of help during the coding process.

Limitations or Misfires: Early versions caused levels to skip because the goal was detected immediately when a new level loaded.

Summary of Process (Human + Tool)

I began by adding a goal to Level 1 and designing a second level that increased difficulty through layout and physics changes. GenAi helped me to understand how a goal would help me achieve my desired outcome. I then added transition and end screens to improve clarity and progression. GenAI was used to help explain how the existing code worked and to support debugging, while all design and implementation decisions were made by me.

Decision Points & Trade-offs

I chose to treat transition and end screens as levels to keep the system fully data-driven. Difficulty was increased by revisiting platform size, spacing, and placement rather than introducing new mechanics. Several platform layouts were adjusted or removed after playtesting to ensure jumps were challenging but still achievable, balancing difficulty with fairness.

Verification & Judgement

I tested the game by playing through the full sequence multiple times. I confirmed that levels loaded in the correct order, gameplay paused on screen levels, and goals triggered only when intended.

Limitations, Dead Ends, or Open Questions

Designing the levels required careful consideration of platform spacing and overall gameplay flow. Some early layouts did not account for how jump distance, player speed, and platform size interacted, which made certain sections feel unplayable. Through iteration, I adjusted platform placement and spacing to better support the intended challenge and improve the overall play experience.

Appendix

https://chatgpt.com/share/69854093-9124-8007-a24a-27bfa800e3e8