Process & Decision Documentation

Side Quest – Celeste (Week 4)

Entry Header

Name:
Elizabeth Ciceu

Role(s):
Designer / Developer (Individual Work)

Primary responsibility for this work:
Designing and implementing a data-driven, array-based platformer level system using p5.js.

Goal of Work Session

The primary goal of this work session was to implement a two-level Celeste-inspired platformer using arrays to generate tile-based levels. Specifically, I focused on debugging player movement, resolving keyboard input issues, and refining Level 2 to ensure it was playable and distinct from Level 1.

Tools, Resources, or Inputs Used

- ChatGPT 5.2 (debugging and structural support)
- Prior p5.js template code
- Lecture materials on arrays and loops
- Course assignment instructions
- Iterative playtesting within the browser
- Previous working version of the sketch file

GenAI Documentation

Date Used:
February 10, 2026

Tool Disclosure:
ChatGPT 5.2

Purpose of Use:
Debugging keyboard input issues, refining level array structure, and modifying level layouts to ensure playability and screen coverage.

Summary of Interaction:
ChatGPT assisted in identifying potential causes of keyboard input failures, restructuring the input handling system, and generating revised Level 2 arrays that maintained difficulty while ensuring beatability. It also helped scale the arrays to better fill the screen dimensions.

Human Decision Point(s):
I rejected multiple suggested input-handling implementations because they overcomplicated the sketch and caused movement to break entirely. I reverted to the last working version of the code and only modified the Level 2 array manually to preserve functional controls. I also adjusted array dimensions myself to ensure levels filled more of the canvas while maintaining balance.

Integrity & Verification Note:
All GenAI suggestions were tested directly in the browser. Any code that interfered with core functionality (movement, collision, dash logic) was removed. I verified that the final implementation aligned with lecture concepts of array-driven level generation and loop-based rendering.

Scope of GenAI Use:
GenAI assisted with debugging logic and generating alternative level arrays. It did not design the full mechanic system, collision system, or dash implementation from scratch. Final balancing decisions and structural adjustments were completed manually.

Limitations or Misfires:
Several GenAI suggestions broke keyboard input entirely or over-engineered the input system. Some generated level arrays were either unbeatable or too similar between levels, requiring manual refinement and playtesting adjustments.

Summary of Process (Human + Tool)

I began with a working Celeste-style movement system and integrated an array-based level structure rendered using nested loops. After implementing Level 2, I discovered it was either unbeatable or the player failed to spawn correctly.

While attempting to fix keyboard input issues, multiple iterations of input handling were tested, including document-level event listeners. These attempts introduced new issues, leading me to revert to the last stable version.

From there, I focused only on modifying the Level 2 array while preserving the working control logic. I expanded both levels to increase screen coverage and redesigned Level 2 to emphasize vertical movement and spikes while maintaining a valid path to the goal.

This process involved repeated playtesting, structural revisions, and balancing adjustments.

Decision Points & Trade-offs

1. Reverting to Stable Controls Instead of Rebuilding Input Logic
I considered fully rewriting the input system using document-level listeners. However, this introduced instability and prevented player movement entirely. I chose to revert to the last working version and limit changes strictly to the level arrays. This preserved functional mechanics while still meeting assignment requirements.

2. Expanding the Level Size
Initially, the arrays were too small and did not visually fill the screen. I considered reducing tile size but chose instead to increase the number of rows and columns. This maintained consistent tile scale while creating a more immersive and complete stage layout.

Verification & Judgement

I evaluated changes through:

- Direct browser playtesting
- Testing reachability of the goal without unintended exploits
- Ensuring spikes functioned as hazards but not hard blockers
- Confirming array-driven generation (no hardcoded tiles)
- Reviewing assignment criteria to ensure loops and arrays were central to implementation

Limitations, Dead Ends, or Open Questions

- Attempts to rewrite keyboard input created instability and required rollback.
- Some larger array layouts initially became too maze-like and confusing.
- Camera movement was not implemented, so very large levels may extend beyond optimal visibility.
- Future improvement could include vertical dash, wall jump, or scrolling camera.

Appendix: GenAI Interaction Transcript

GBDA 302 – Side Quest Week 4 (Celeste Platformer)
Assignment: Side Quest Week 4
Nature of Work: Individual
Tool Referenced: ChatGPT 5.2

Purpose of Appendix

This appendix documents how generative AI was used as a support tool while developing the Celeste-inspired platformer side quest. The interaction focused on debugging, level balancing, and expanding array-based level structures, rather than generating the entire project from scratch.

The transcript demonstrates how the student used GenAI to troubleshoot keyboard input issues, refine level arrays for playability, and expand level dimensions to better fill the canvas. Throughout the process, the student maintained control over movement mechanics, collision logic, and final design decisions.

Transcript Summary (Edited for Clarity)

Student:
What's Side Quest 4 for this week?

Tool Response:
Provided a breakdown of the assignment options and suggested building a Celeste-style platformer using arrays and loops for level generation.

Student:
I think I want to do Celeste.

Tool Response:
Confirmed that a Celeste-style platformer would align well with the assignment and suggested building from a starter p5.js example using array-driven tile maps.

Student:
There's something wrong, when you reach level 2, the blob disappears. Also, the red exit doesn't restart the game. I think the for loop is wrong.

Tool Response:
Suggested reviewing spawn logic and level transition handling. Proposed restructuring level loading and ensuring spawn positions were recalculated when switching arrays.**Student:**
Level 2 is unbeatable, can you make it beatable?

Tool Response:
Generated a revised Level 2 array with a clearer route to the goal and fewer blocking spike placements.

Student:
Can I have the whole code?

Tool Response:
Provided a full sketch including start screen, win screen, dash mechanic, and two array-based levels.

Student:
The arrow keys aren't working.

Tool Response:
Suggested preventing default browser scrolling behavior and modifying keyboard event listeners to capture input reliably.

Student:
Now none of the keys work.

Tool Response:
Proposed a more robust input-handling approach using `window.addEventListener` and storing both `e.code` and `e.key`values.

Student:
Now I can't move the blob at all.

Tool Response:
Diagnosed that the revised input system may have overcomplicated the working logic and recommended reverting to the last stable version.

Student:
This is the last code that had working controls. Only change Level 2 to make it beatable.

Tool Response:
Modified only the Level 2 array while preserving all existing movement and collision logic.

Student:
Can you make both arrays bigger to fill the whole screen, and make sure Level 1 and Level 2 are different?

Tool Response:
Generated larger (24x16) level arrays and differentiated them structurally:

- Level 1 emphasized horizontal routing and maze-like structure.
- Level 2 emphasized vertical platforming and spike hazards.

Student:
Can we do the process document following the provided format?

Tool Response:
Drafted a structured Process & Decision document aligned with the course template, including a GenAI Documentation section and reflection components.

Summary of GenAI Use

GenAI was used to:

- Troubleshoot spawn and level transition issues
- Suggest alternative input-handling strategies
- Generate revised and expanded level arrays
- Assist in drafting structured documentation

All final decisions regarding:

- Movement mechanics
- Collision detection
- Dash behavior
- Final array selection
- Level balancing
- Reverting broken input systems

were made by the student.

Several AI-suggested input rewrites were rejected after testing because they interfered with player movement. The student ultimately reverted to a stable version and only modified level arrays manually to preserve functionality.

GenAI functioned as a debugging and iteration support tool rather than the primary author of the project.