

Process & Decision Documentation

Project/Assignment Decisions

Pivot from a simple "reach the goal" mechanic to a feature-rich platformer with collectibles, moving platforms, and a scoring system.

Entry Header

Name: Nicole Wu

Role(s): Game Designer / Programmer

Primary responsibility for this work: Developing a scalable JSON-driven platformer with progressive difficulty and polished visuals.

Goal of Work Session

To create a 5-level game engine where levels are loaded dynamically from an external JSON file.

Tools, Resources, or Inputs Used

GitHub Copilot, VS Code, Python (local server), JSON.

GenAI Documentation

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

Date Used: 2026-02-10

Tool Disclosure: GitHub Copilot

Purpose of Use: Generative coding for class structures (Player, Level), debugging physics (jump height), and creating JSON data for multiple levels.

Summary of Interaction: The AI generated a reusable class-based engine and progressively difficult levels while assisting in the refactoring of a HUD and particle system.

Human Decision Point(s): I overruled the AI's difficulty settings and visual styles to ensure Level 3 was playable and the UI more aesthetic.

Integrity & Verification Note: I manually tested each level after generation to ensure the difficulty curve was fair and that the "auto-load" feature worked correctly between levels.

Scope of GenAI Use: Copilot wrote the boilerplate code and JSON data; I defined the gameplay rules (score penalty on fall, +5 on win) and fine-tuned the user experience.

Limitations or Misfires: The tool initially failed to set a canvas size and placed platforms in unreachable locations, requiring manual tuning.

Summary of Process (Human + Tool)

- I tasked Copilot with setting up a JSON-based engine to solve the "hard to reuse" code problem.
- We tested Level 1, found the jump physics lacking, and tweaked the velocity.
- I pushed for "creative features," leading to the addition of enemies and moving platforms.
- I play tested Level 3, found it frustrating, and had the AI re-balance the platform layout.
- We moved from basic-colored rectangles to a polished visual style with particle effects.

Decision Points & Trade-offs

- Implementing a "Score Penalty" system (-1 for falling, +5 for winning).
- The game shifted from a simple "completionist" loop to a "high score" challenge.

Verification & Judgement

- I ran the game on a local server to verify that the JSON file loaded correctly and that the "Next Level" trigger fired reliably.
- I confirmed that the HUD and instructions were visible within the canvas boundaries on different screen sizes.

Limitations, Dead Ends, or Open Questions

- The AI-generated levels sometimes placed platforms too far apart, requiring manual "tuning" prompts to make them playable for an average user.

Appendix

User: Generate a level using arrays or JSON data; use loops to dynamically place tiles, obstacles, or platforms.

- Bonus: Add a second level that loads automatically when the first is finished. This is the objective: Three Problems
- Spaghetti Code: All of your code in draw() with no clear structure
- Repeated Logic: Drawing the same kind of thing three different ways.
- Hard to reuse: If you want five levels, does it make sense to copy/paste the code five times?

JSON & Classes

- JSON files: "I have similar data (like multiple levels) that I want to load at runtime without rewriting code."
- Classes: "I have a concept (like 'a level' or 'a player') that bundles data + behaviour together,
and I want to reuse it."

GitHub Copilot: I'll create a small HTML + JS project with JSON levels and classes, then wire automatic level loading. I'll add `index.html`, `js/game.js`, `data/levels.json`, and update README.md.

Created 3 todos

Now I'll add the project files and implementation. I'll read the existing README.md to append run instructions.

Read

[](file:///c%3A/Users/34995/Downloads/UW%20Classworks/GBDA302/n9wu_sidequest_W4/README.md)

Now I'll add the files (`index.html`, `js/game.js`, `data/levels.json`) and update README.md.

Using "Apply Patch"

```

```

Using "Apply Patch"

```

```

Using "Apply Patch"

```

```

Using "Apply Patch"

```

```

Completed: *Create project files* (1/3)

Progress: Created the demo files and implemented JSON-driven levels with classes; next step is to run it in a browser.

What I added

- index.html — simple canvas page. See index.html
- game.js — `Player`, `Platform`, `Obstacle`, `Level`, `Game` classes; level loading and auto-advance. See game.js
- levels.json — two level definitions (platforms, obstacles, start, goal). See levels.json
- Updated README.md with run instructions. See README.md

How to try it

- From the project folder run:

```
```bash
python -m http.server 8000
```

```

- Open <http://localhost:8000> in a browser and play (Arrow keys + Space).

Want me to (pick one)

- Commit the files to git
- Add a third level or hazards (moving enemies)
- Improve collisions or add a score/goal counter

Made changes.

User: what kind of game is this, explain the game mechanic

- GitHub Copilot:
- **Type**: 2D single-screen platformer demo (jump-and-run style).
 - **Goal**: Move the player to the green goal box to finish a level; completing level 1 automatically loads level 2, finishing all levels shows a win message.
 - **Mechanics**: gravity + jumping, horizontal movement, AABB collision detection with platforms and obstacles; hitting an obstacle resets the player to the level start.
 - **Levels / Data**: level layouts (platforms, obstacles, player start, goal) are stored in `levels.json` and loaded at runtime so you can add levels without changing game code.
 - **Architecture**: reusable classes (`Player`, `Platform`, `Obstacle`, `Level`, `Game`) implemented in `game.js` keep logic out of a single `draw()` and make expansion easy.
 - **Controls**: `ArrowLeft`, `ArrowRight` to move; `Space` or `ArrowUp` to jump.

User: when i click jump i cant reach the green box

GitHub Copilot: I'll add a short todo and inspect the player physics to find the jump impulse.

Created 3 todos

Read

[](file:///c%3A/Users/34995/Downloads/UW%20Classworks/GBDA302/n9wu_sidequest_W4/js/game.js)

I'll patch game.js to increase the player's jump velocity so jumps reach the goal.

Using "Apply Patch"

```

```

Starting: *Test and iterate* (3/3)

Change made: increased jump impulse in game.js (vy from -320 to -420) so the player can reach higher goals. Try it by serving the folder and opening the page:

``` bash

python -m http.server 8000

```

Then open <http://localhost:8000> and test the jump. Want me to tune jump/gravity further or add a double-jump option?

Made changes.

User: can u add more level after reach the level 2

GitHub Copilot: I'll add a third level to levels.json and update the todo list.

Starting: *Add extra level(s)* (1/3)

Using "Apply Patch"

I'll read levels.json to get current content before patching.

Read

```
[](file:///c%3A/Users/34995/Downloads/UW%20Classworks/GBDA302/n9wu_sidequest_W4/data/levels.json)
```

Now I'll append a third level to levels.json by updating the file.

Using "Apply Patch"

...

```

Starting: \*Test locally\* (2/3)

Done — I added a third level to levels.json. Progress: level added (testing in progress).

To test locally:

``` bash

python -m http.server 8000

```

Open <http://localhost:8000>, finish level 2 and the game will load the new level automatically.

Want me to also update README.md to note 3 levels or add a fourth level?

Made changes.

User: The level 3 is so difficult i cant even pass, can u make a little bit arrangement

GitHub Copilot: I'm going to relax Level 3: widen platforms, add a safer path, and lower the goal so it's reachable. I'll update levels.json.

Read

[](file:///c%3A/Users/34995/Downloads/UW%20Classworks/GBDA302/n9wu\_sidequest\_W4/data/levels.json)

Created 4 todos

Using "Apply Patch"

```

```

Completed: \*Adjust Level 3 difficulty\* (4/4)

Done — I relaxed Level 3 by widening platforms, removing extra hazards, and lowering the goal so it's reachable.

Quick test:

``` bash

python -m http.server 8000

```

Open <http://localhost:8000>, finish level 2 and try Level 3. Want further tweaks (lower goal more, add a mid-platform checkpoint, or add a visual label)?

Made changes.

User: ok, can u add more feactures in it, be creative and more interesting

GitHub Copilot: I'll first add a brief plan to the todo list, then patch game.js to add collectibles, moving platforms, enemies, double-jump, and HUD; finally update levels.json to include coins and a moving platform in level 3.

Starting: \*Plan features\* (1/5)

Using "Apply Patch"

```

```

Read

[]([file:///c%3A/Users/34995/Downloads/UW%20Classworks/GBDA302/n9wu\\_sidequest\\_W4/data/levels.json](file:///c%3A/Users/34995/Downloads/UW%20Classworks/GBDA302/n9wu_sidequest_W4/data/levels.json))

Using "Apply Patch"

```

```

Starting: \*Test locally\* (4/5)

Progress update: I added creative features — double-jump, collectibles (coins), moving platforms, simple patrolling enemies, and a HUD; and I updated Level 3 to include those elements.

### What changed

- game.js: Added `Collectible`, `MovingPlatform`, `Enemy` classes; double-jump support; integrated moving platforms/enemies/coins into update/draw logic; added score HUD.
- levels.json: Level 3 now contains `movingPlatforms`, `coins`, and `enemies`.

### How to test

```
``` bash
python -m http.server 8000
````
```

Open <http://localhost:8000>, collect coins for points, use double-jump (Space/Up twice) to reach moving platforms, avoid enemies, and complete levels.

### Next options (pick one)

- I can tune values (platform ranges/speeds, coin placement, enemy speed) for balance.
- Add sounds, particle effects, or checkpoints.
- Update README.md with the new features.

### Made changes.

User: ok, for the score can u make it when u drop once, u minus 1 point, pass 1 level add 5 points

GitHub Copilot: I'll update the game's scoring: decrement on resets (obstacle/fall/enemy) and add 5 points when a level is passed. I'll patch game.js accordingly.

Using "Apply Patch"

Read

[](file:///c%3A/Users/34995/Downloads/UW%20Classworks/GBDA302/n9wu\_sidequest\_W4/js/game.js)

Using "Apply Patch"

```

```

Updated todo list

Done — scoring updated:

- Level complete: +5 points (added in `nextLevel()`).
- Reset by obstacle, enemy, or falling: -1 point (applied where resets occur).
- Coins still give +10 each.

Test locally:

```bash

```
python -m http.server 8000
```

```

Open <http://localhost:8000> and verify scoring behavior. Want me to clamp the minimum score to 0 or show score change popups?

Made changes.

User: good, now can u make it more visually appealing

GitHub Copilot: I'll add a short plan then patch index.html and game.js to improve visuals (background, shadows, rounded player, goal glow, coin shine, HUD styling, simple particles).

Starting: \*Visual overhaul plan\* (1/4)

Read

[]([file:///c%3A/Users/34995/Downloads/UW%20Classworks/GBDA302/n9wu\\_sidequest\\_W4/index.html](file:///c%3A/Users/34995/Downloads/UW%20Classworks/GBDA302/n9wu_sidequest_W4/index.html))

Using "Apply Patch"

```

```

Using "Apply Patch"

```

```

Using "Apply Patch"

```

```

Starting: \*Test visuals locally\* (4/4)

I've updated visuals: background gradient and parallax clouds, rounded player, platform highlights, glowing goal, shiny coins, particle effect on coin collect, moving-platform visuals, and a styled HUD with rounded background.

Next: run the demo locally to verify performance:

``` bash

python -m http.server 8000

```

Open <http://localhost:8000>. Would you like me to add ambient music, particle effects for level completion, or polish the UI colors?

Made changes.

User: add two more levels, and the difficulty should be progressively increase

GitHub Copilot: I'll add two progressively harder levels to levels.json and update the todo list.

Added 3 todos

Using "Apply Patch"

```

```

Starting: \*Test new levels locally\* (6/7)

Done — I added Level 4 and Level 5 to levels.json with progressively increased difficulty (narrower platforms, faster/more moving platforms, more enemies/obstacles). Test locally:

``` bash

```
python -m http.server 8000
```

```

Open <http://localhost:8000>, play through levels — difficulty should ramp up across levels 1 → 5.

If you'd like, I can:

- Tune balance (speeds, ranges, enemy counts)
- Clamp score to not go negative
- Add level-select UI or a visual indicator for difficulty per level. Which would you like next?

Made changes.

User: for the game instruction: Arrow keys to move · Space to jump · Double-jump enabled.  
is out of the canva, can u put it inside the canva?

GitHub Copilot: I'll remove the external footer from index.html and draw the instructions inside the canvas by updating game.js's HUD drawing.

Using "Apply Patch"

```

```

Using "Apply Patch"

Read

[](file:///c%3A/Users/34995/Downloads/UW%20Classworks/GBDA302/n9wu\_sidequest\_W4/js/game.js#1-1), lines 1 to 400

Read

[](file:///c%3A/Users/34995/Downloads/UW%20Classworks/GBDA302/n9wu\_sidequest\_W4/js/game.js#400-400), lines 400 to 446

Using "Apply Patch"

```

```

Done — the external footer was removed and the instruction text is now drawn inside the canvas (bottom center). Test with:

``` bash

python -m http.server 8000

```

Open <http://localhost:8000>. Want the instructions styled differently, or moved to the top/side inside the canvas?

Made changes.