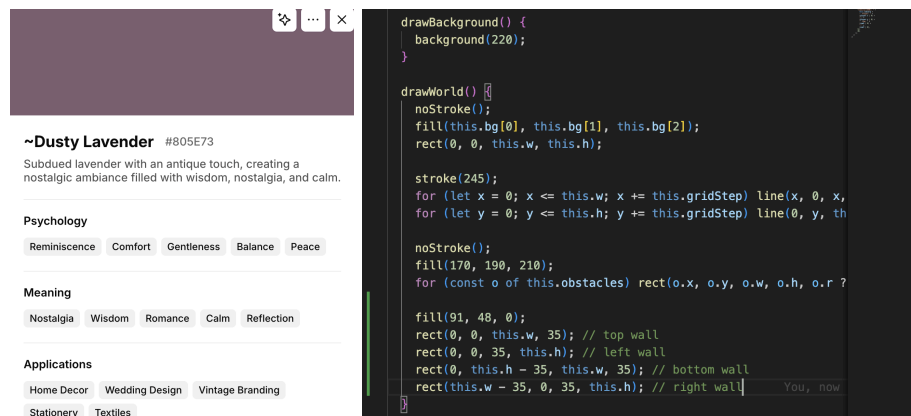


Process & Decision Documentation

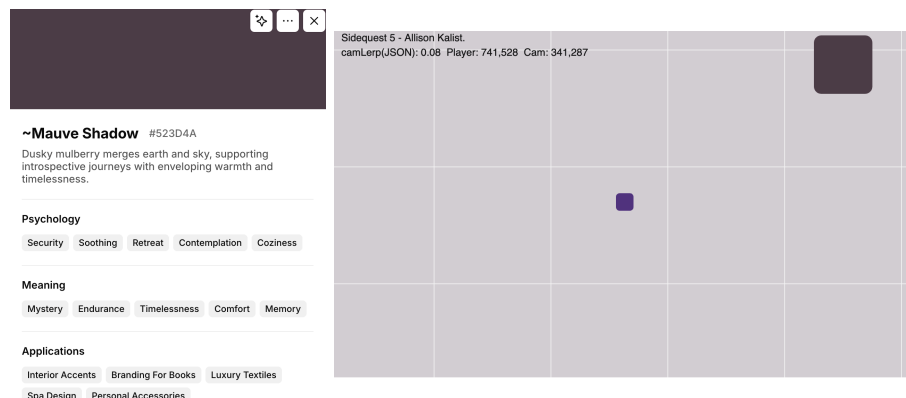
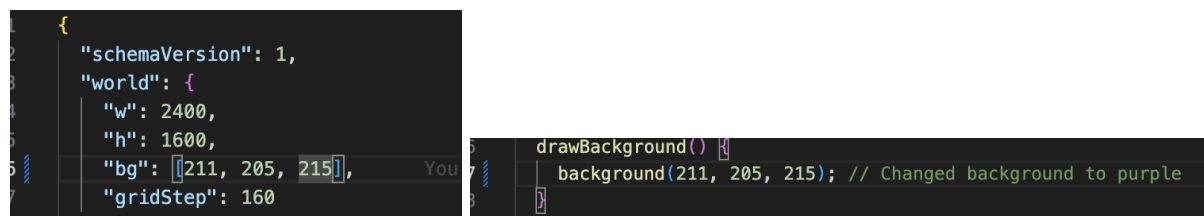
Project/Assignment Decisions

For this sidequest I wanted to try incorporating a mix of my own coding but also using GenAI for assistance. I knew I'd be able to change colours on my own and add more obstacles, which I was, but I also wanted to increase the player speed and add a shake to the camera, which I knew I would need help with. Documentation below.

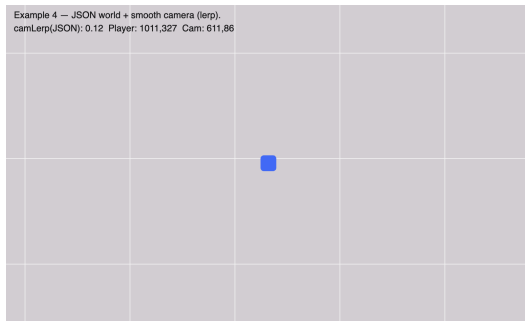
Below is the documentation of my own edits and the personal decisions that I made. These screenshots and annotations explain my thought process for the design and how I manipulated the code myself.



Annotation: I started by adding “walls” to the map so that it’s more obvious where the world borders are.



Annotation: I also changed the background colour, the obstacles, and the player to purple for “fear”.



```

    "obstacles": [
      { "x": 40, "y": 40, "w": 80, "h": 80, "r": 10 },
      { "x": 1000, "y": 300, "w": 80, "h": 80, "r": 60 }
    ]
  }
  You, now * Uncommitted changes
}

```

Annotation: I added in extra obstacles. I was struggling with having them appear, so I went back to the example 3 code which has multiple obstacles, and copied the call function in [sketch.js](#). You can see in the above screenshots that my player was at the location that the obstacle should have been, but it wasn't appearing.

Below is the copied code from example 3 in WorldLevel.js, and the finalized obstacle code in world.json.

```

noStroke();
fill(82, 61, 74); // update obstacle colour
// for (const o of this.obstacles) rect(o.x, o.y, o.w, o.h, o.r ?? 0);
// Took this code from example 3 to generate more obstacles
for (const o of this.obstacles) {
  rect(o.x, o.y, o.w, o.h, o.r ?? 0);
}

```

```

    "obstacles": [
      { "x": 40, "y": 40, "w": 80, "h": 80, "r": 10 },
      { "x": 1000, "y": 300, "w": 80, "h": 80, "r": 10 },
      { "x": 79, "y": 1249, "w": 80, "h": 80, "r": 10 },
      { "x": 500, "y": 137, "w": 80, "h": 80, "r": 10 },
      { "x": 925, "y": 868, "w": 80, "h": 80, "r": 10 },
      { "x": 1980, "y": 1140, "w": 80, "h": 80, "r": 10 },
      { "x": 1830, "y": 502, "w": 80, "h": 80, "r": 10 },
      { "x": 1500, "y": 195, "w": 80, "h": 80, "r": 10 },
      { "x": 420, "y": 780, "w": 80, "h": 80, "r": 10 },
      { "x": 675, "y": 1320, "w": 80, "h": 80, "r": 10 },
      { "x": 217, "y": 547, "w": 80, "h": 80, "r": 10 },
      { "x": 1395, "y": 1200, "w": 80, "h": 80, "r": 10 },
      { "x": 1597, "y": 757, "w": 80, "h": 80, "r": 10 }
    ]
  }
}

```

```

"playerStart": {
  "x": 1200,
  "y": 800,
  "speed": 17
},

```

Annotation: I increased the player speed to enhance the feel of "fear".

```

function draw() {
  player.updateInput();

  // The following code was taken from ChatGPT
  const moving =
    keyIsDown(LEFT_ARROW) ||
    keyIsDown(RIGHT_ARROW) ||
    keyIsDown(UP_ARROW) ||
    keyIsDown(DOWN_ARROW) ||
    keyIsDown(65) ||
    keyIsDown(68) ||
    keyIsDown(87) ||
    keyIsDown(83);

  if (moving) {
    triggerFear(0.12);
  }

  //Return to original code
}

```

Annotation: I wanted to add a “shake” to the camera when the player moves to further convey the sense of fear. I used ChatGPT to help me generate the original code but I did have to make some tweaks to make it better fit what I wanted (documented below).

```
"camera": {  
  "lerp": 0.08,  
  "fearShake": {  
    "max": 6,  
    "decay": 0.9,  
    "frequency": 0.15  
  }  
}
```

(original GenAI values)

```
"camera": {  
  "lerp": 0.08,  
  "fearShake": {  
    "max": 15,  
    "decay": 0.5,  
    "frequency": 0.7  
  }  
}
```

(my edited values)

Increased “max” value, increased “frequency”, and decreased “decay”.

```
if (moving) {  
  triggerFear(9);  
}
```

Increased “triggerFear” value (amount)

```
// The following code was taken from ChatGPT  
function triggerFear(amount) {  
  shakeAmount = min(shakeAmount + amount, level.fearShake.max);  
}  
// Return to original code
```

GenAI Documentation

Date Used: Thursday, February 12, 2026

Tool Disclosure: ChatGPT v 5.2

Purpose of Use: I used GenAI exclusively to generate the code for the camera shake and to help me figure out why it wasn’t working initially.

Summary of Interaction: I pasted in four of the pages of code so that it could understand what already existed, and I asked it to generate code that would add a camera shake. It gave me a step-by-step breakdown of all the code I needed to add, and where specifically I needed to add it. When I said it wasn’t working, it told me that the camera shake was probably just very subtle, so I then increased the values.

Human Decision Point(s): The main decisions that I made with the GenAI code was that I updated the numbers in the code that GenAI gave me. The numbers that it gave me were too small for there to be a noticeable shake in the camera, but I wanted it to be really noticeable so I increased all the numbers.

The AI and I also discovered that there was a problem with the player speed because I hadn’t originally pasted the [Player.js](#) file, so ChatGPT didn’t know how the speed was being tracked. Because of this, it had given me the wrong code for tracking velocity.

Once I pasted the [player.js](#) file, I had to make sure to delete the old line of code, which Chat did not tell me to do, so I had to solve this on my own.

It also wanted me to run a bunch of tests to check the camera shake but I didn't want to alter the code too much, so I just edited the number values and that got to my desired effect.

Integrity & Verification Note: I made notes of where I added in the GenAI curated code, and I made sure not to delete any existing code, in case the new code didn't work. When I didn't see any noticeable camera shake, I asked the AI why this would be happening and it said that it was likely just because the shake was very subtle.

Scope of GenAI Use: I documented the process of my own work above but in summary: GenAI ONLY created the camera shake and the original code associated with it. My original code included changing colours, adding visible walls to the world borders, adding in more obstacles, and altering the numbers for the camera shake so that it was more obvious.

Limitations or Misfires: The tool generated the original code for the camera shake very well, but when I wasn't seeing any improvement I asked for the reason why and it told me, but it wanted me to do a bunch of tests that would have messed with the code. It didn't even tell me that I could have just adjusted the numbers, I had to figure this out myself. It did a good job of telling me that the camera shake was subtle and this was why I wasn't seeing it, but it would have taken me down a long line of testing and screwing with the code that might have messed up my code even more.

Summary of Process (Human + Tool)

This process is documented in more detail above with my screenshots, but here is a summary.

1. Generate the walls on the world border (original)
2. Change colours for obstacles, background, walls, and player (original)
3. Generate more obstacles with assistance from example 3 code (original)
4. Increase player speed (original)
5. Add camera shake (ChatGPT generated)
6. Increase camera shake (my edits)

Decision Points & Trade-offs

1. I mentioned this previously but I decided to edit the number values in the camera shake so that it was more visible. I wanted the camera to give off the feeling of “fear”, and the GenAI generated code was much too subtle.

Verification & Judgement

I tested with multiple sets of numbers to see which ones matched the camera shake that I was trying to go for. This trial-and-error method helped me figure out if the code itself was correct and just needed the values to be different, which ended up being the problem.

Limitations, Dead Ends, or Open Questions

I’ve noted this before, but I had to “compromise” with how to make sure the shake was working. The AI wanted me to add a bunch of additional code, but it also told me that the problem was that the shake wasn’t very visible, so I figured out that I just needed to change the numbers, and that worked. I’ve noticed this a lot with genAI that it often does not provide the easiest solution, and instead leads you somewhere you don’t want to go. I’m glad I had the coding experience to recognize that this extra code was not necessary; it saved me a lot of trouble.

Appendix

Please include a full transcript of your conversation with the GenAI. GenAI transcripts are used for transparency and verification only and are not assessed for writing quality or completeness.

Note: If GenAI is used in another language, you must include both the original language and a translated transcript in the appendix

Me = purple

GenAI = Green

I'm going to paste two .js files and one JSON file... I want you to tell me how to make the camera shake to convey a sense of "fear" for the user.

Sounds good — go ahead and paste the **two .js files** and the **JSON file**.

Once I can see:

- how the camera is created/updated,
- what engine or framework you're using (e.g., Three.js, Babylon.js, Phaser, custom WebGL),
- and how your game loop / state is structured,

I'll give you **specific, code-level changes** to add camera shake that conveys *fear* (e.g., subtle jitter vs. panic shake, timing, decay, randomness, and player-state triggers), not just a generic recipe.

Paste the files when ready 👍

Pasted [sketch.js](#), [world.json](#), and [WorldLevel.js](#)

Awesome — with all three files, we can add **fear-based camera shake** cleanly and *data-driven*, without ruining your smooth follow.

Below is a **layered approach** that fits GBDA302 design goals and reads well if you have to explain it in critique.

Concept: “Fear” camera vs “Action” camera

Your camera already does this well:

- **Lerp follow** → calm, controlled movement

We'll add:

- **Micro-instability** layered *after* lerp
- **Low amplitude**
- **Irregular motion**
- **Slow decay**

This makes the player feel *uneasy*, not excited.

1 Add shake parameters to world.json

This keeps everything **tunable without touching code**.

```
"camera": {  
  "lerp": 0.08,  
  "fearShake": {  
    "max": 6,  
    "decay": 0.9,  
    "frequency": 0.15  
  }  
}
```

What these mean (design-friendly):

- max: how scared the camera can get
 - decay: how quickly fear fades (closer to 1 = longer dread)
 - frequency: how “nervous” the shake feels
-

2 Read shake data in WorldLevel.js

Add this inside the constructor:

```
this.fearShake = {  
  max: json.camera?.fearShake?.max ?? 4,  
  decay: json.camera?.fearShake?.decay ?? 0.92,  
  frequency: json.camera?.fearShake?.frequency ?? 0.15  
};
```

Now the **level defines emotional tone**, not the player.

3 Add shake state to sketch.js

At the top of sketch.js:

```
let shakeAmount = 0;  
let shakeTime = 0;
```

Add a helper function (below setup() is fine):

```
function triggerFear(amount) {  
  shakeAmount = min(shakeAmount + amount, level.fearShake.max);  
}
```

This lets *anything* cause fear later (dark zones, sounds, enemies).

4 Apply shake AFTER camera lerp (this is critical)

In draw(), **after** this block:

```
camX = lerp(camX, targetX, camLerp);  
camY = lerp(camY, targetY, camLerp);
```

Add:

```
// Fear-based camera shake  
if (shakeAmount > 0.01) {  
  shakeTime += level.fearShake.frequency;  
  
  const offsetX = noise(shakeTime) * 2 - 1;  
  const offsetY = noise(shakeTime + 100) * 2 - 1;  
  
  camX += offsetX * shakeAmount;  
  camY += offsetY * shakeAmount;
```



```
shakeAmount *= level.fearShake.decay;  
}
```

Why this works

- `noise()` → organic, breathing motion
 - different noise seeds → asymmetry (more unsettling)
 - `decay` → fear lingers, then fades
-

5 Trigger fear (simple example)

For now, trigger fear when the player moves fast:

```
if (abs(player.vx) + abs(player.vy) > 0) {  
    triggerFear(0.15);  
}
```

Or on reset:

```
if (key === "r" || key === "R") {  
    triggerFear(4);  
}
```

Later, you can trigger it when:

- entering dark spaces
- approaching obstacles
- standing still too long
- hearing audio cues

Pasted the [Player.js](#) code to see if it would make a difference

Yes — **this *does* change things**, and you caught it at exactly the right moment 👍
Your Player **does NOT track velocity**, so the earlier `player.vx / player.vy` example would **not work**.

That's totally fine — it just means we trigger fear **based on input**, not velocity.

Below is the **correct, clean solution for *your* Player.js**.

Why your Player.js changes the approach

Your player:

- moves **directly** by input each frame
- does **not** store `vx`, `vy`, or movement state
- normalizes diagonal movement using `1/en`

So:

- there is **no velocity to read**
- but input itself is still a perfect fear signal

This is actually **very appropriate for psychological fear**.

✅ Correct fear trigger for *your* code

Put this inside `draw()` in `sketch.js`

Right after:

```
player.updateInput();
```

Add:

```
// Fear increases while the player is actively moving
const moving =
  keyIsDown(LEFT_ARROW) || keyIsDown(RIGHT_ARROW) ||
  keyIsDown(UP_ARROW) || keyIsDown(DOWN_ARROW) ||
  keyIsDown(65) || keyIsDown(68) ||
  keyIsDown(87) || keyIsDown(83);

if (moving) {
  triggerFear(0.12);
}
```

That's it. No changes to `Player.js` required.

The camera does not appear to be shaking at all, is any of the previous code not supposed to be included?

Good news first: **nothing you added is “wrong”**, but there *is* a very common p5.js gotcha here, plus one tuning issue that can make the shake effectively invisible.

I'll walk you through this **systematically** and give you a **guaranteed-to-work test** so you can see it immediately.

✅ What should still be included (nothing to remove)

You **should keep all of this**:

- the lerped camera (`camX = lerp(...)`)
- the shake state (`shakeAmount`, `shakeTime`)
- the `noise()`-based offset

- the `triggerFear()` function
- the input-based fear trigger in `draw()`

So no — you didn't include something you shouldn't have 👍

! Why you're not seeing shake (the real reasons)

1 Your shake is probably too subtle to notice

Right now:

- offsets are very small
- decay happens *every frame*
- lerp smoothing dampens movement even more

So the shake **exists**, but it's below perception.

This is *especially* true on a grid-based background where motion needs to cross pixel boundaries to read.

2 You don't yet have a visual sanity check

Before tuning "fear", we need to **prove the system works**.

So we'll do a temporary exaggeration test.

🔥 Step 1: Add a HARD TEST TRIGGER (do this first)

In `sketch.js`, add this to `keyPressed()`:

```
function keyPressed() {
```

```

    if (key === "r" || key === "R") {
      const start = worldData.playerStart ?? { x: 300, y: 300,
speed: 3 };
      player = new Player(start.x, start.y, start.speed);
    }

    // 🔥 TEST: force camera shake
    if (key === "f" || key === "F") {
      triggerFear(20);
    }
  }
}

```

Now press **F** while the sketch is running.

- 👉 If the camera shakes after this step, the system works and we just need tuning.
- 👉 If it *still* doesn't shake, continue to Step 2.

Step 2: Temporarily exaggerate the shake math

Find this code in `draw()`:

```

if (shakeAmount > 0.01) {
  shakeTime += level.fearShake.frequency;

  const offsetX = noise(shakeTime) * 2 - 1;
  const offsetY = noise(shakeTime + 100) * 2 - 1;

  camX += offsetX * shakeAmount;
  camY += offsetY * shakeAmount;

  shakeAmount *= level.fearShake.decay;
}

```

Replace it temporarily with this exaggerated version:

```
if (shakeAmount > 0.01) {  
    shakeTime += 0.5; // BIG step for visibility  
  
    const offsetX = random(-1, 1);  
    const offsetY = random(-1, 1);  
  
    camX += offsetX * shakeAmount;  
    camY += offsetY * shakeAmount;  
  
    shakeAmount *= 0.85; // slower decay  
}
```

Now press **F** again.

What you should see

- The entire world visibly jittering
- The grid “buzzing”
- A very obvious shake

If you see this → ☒ the pipeline is 100% correct.

If it shakes now (most likely outcome)

That means:

- nothing was broken
- the original values were just *too subtle*

You can now safely return to the **fear-appropriate** version, but with **stronger values**.

🎯 Final tuned version (balanced fear)

Use this as your final shake block:

```
if (shakeAmount > 0.01) {
  shakeTime += level.fearShake.frequency;

  const offsetX = (noise(shakeTime) - 0.5) * 2;
  const offsetY = (noise(shakeTime + 1000) - 0.5) * 2;

  camX += offsetX * shakeAmount;
  camY += offsetY * shakeAmount;

  shakeAmount *= level.fearShake.decay;
}
```

And update world.json to **stronger but still subtle fear**:

```
"fearShake": {
  "max": 8,
  "decay": 0.88,
  "frequency": 0.25
}
```

Can I just change the value in the triggerFear() function?

Yes — **you absolutely can**, and in your case that's the *simplest* fix 👍

If the shake isn't visible, the most likely reason is just that this number is too small:

```
triggerFear(0.12);
```

That value is extremely subtle, especially with:

- lerp smoothing (0.08)

- decay happening every frame
- small noise offsets