Process & Decision Documentation

Project/Assignment Decisions

Side Quests and A4 (Individual Work)

I changed the project into a reflective exploration experience. I made this change because the camera and player movement felt more meditative than goal-based gameplay. This helped create a calmer and more emotional interaction.

GenAI Documentation

Date Used: Feb 24, 2026

Tool Disclosure: ChatGPT 5.2

Purpose of Use: I used GenAI to help debug code and improve the camera movement and visual design.

Summary of Interaction: The tool suggested ways to make the camera drift, add hidden symbols, and improve the player design.

Human Decision Point(s): I removed some effects like heavy borders because they blocked the view. I also adjusted the visuals to match the assignment goal.

Integrity & Verification Note: I tested all changes myself to make sure they worked and matched the reflective experience requirement.

Scope of GenAI Use: GenAI helped with ideas and code refinement, but I made final design decisions and adjustments.

Limitations or Misfires: Some suggestions made the screen unclear or removed movement, so I had to fix and simplify them.

Summary of Process (Human + Tool)

I tested different camera styles and visual effects. Some ideas did not work and needed to be changed. I kept adjusting until the movement felt calm and smooth.

Decision Points & Trade-offs

I chose to remove goals and focus on exploration. I also replaced hard borders with soft environmental edges to improve mood.

Verification & Judgement

I checked the experience by playing it myself. I made sure the movement felt slow and reflective.

Limitations, Dead Ends, or Open Questions

Some visual effects looked good but blocked the player. I had to simplify them to keep the experience clear.

Appendix

1st Iteration:

[insert Javascript and json files] based on what I have so far, update and revise the code as needed (include text explanation) to meet this goal: Create a reflective or meditative camera experience that scrolls through a world larger than the screen. Use pacing and motion to evoke emotion

1st Response:

Great — your current system already **technically works** as a functional camera-world setup.

But right now it feels like a *game camera*.

Your assignment goal is different:

> ❝ Create a reflective or meditative camera experience that scrolls through a world larger than the screen. Use pacing and motion to evoke emotion ❞

So we need to shift from:

👉 **Control + precision**
to
👉 **Drift + atmosphere**

Below is a **conceptual upgrade + exact code changes**.

🎯 Core Design Shift

Right now:

- Camera is locked tightly to player center
  sketch
- Player moves instantly via direct input
  Player
- Motion is mechanical

This creates urgency.

Meditation needs:

✔ Slow response
✔ Soft lag
✔ Floating movement
✔ Space that feels bigger than control

So we introduce:

🌫️ Camera inertia
🌊 Player momentum
🫧 Breathing motion

✨ 1. Transform Player → Drifting Presence

Update **Player.js**

Replace `updateInput()` with:

```
updateInput() {
 const ax =
   (keyIsDown(RIGHT_ARROW) || keyIsDown(68)) -
   (keyIsDown(LEFT_ARROW) || keyIsDown(65));
```

```
 const ay =
   (keyIsDown(DOWN_ARROW) || keyIsDown(83)) -
   (keyIsDown(UP_ARROW) || keyIsDown(87));

 // introduce velocity + easing
 this.vx = lerp(this.vx ?? 0, ax * this.s, 0.05);
 this.vy = lerp(this.vy ?? 0, ay * this.s, 0.05);

 this.x += this.vx;
 this.y += this.vy;
}
```

👉 Now movement has **lag and glide**

No more robotic stepping.

✨ 2. Make Camera Follow Emotionally (not mechanically)

Right now camera snaps to player center

sketch

Replace camera section in **sketch.js**

Find:

```
camX = constrain(player.x - width / 2, 0, maxCamX);
camY = constrain(player.y - height / 2, 0, maxCamY);
```

Replace with:

```
let targetX = constrain(player.x - width / 2, 0, maxCamX);
let targetY = constrain(player.y - height / 2, 0, maxCamY);
```

// smooth emotional drift

camX = lerp(camX, targetX, 0.03);

camY = lerp(camY, targetY, 0.03);

👉 Camera now **follows slowly**

Creates contemplative delay.

✨ 3. Add Breathing Motion (Subtle Life)

Still in **draw()** after updating cam:

// subtle breathing drift

let breathX = sin(frameCount * 0.01) * 8;

let breathY = cos(frameCount * 0.008) * 6;

camX += breathX;

camY += breathY;

Now the world gently moves even when player stops.

This is crucial for:

🧘 Stillness without stagnation

✨ 4. Expand World Atmosphere

Your world already exceeds the screen

world

But visually it still feels grid-like and technical

WorldLevel

So soften it.

Update `drawWorld()` in **WorldLevel.js**

Replace grid section:

```
stroke(245);
for (let x = 0; x <= this.w; x += this.gridStep) line(x, 0, x, this.h);
for (let y = 0; y <= this.h; y += this.gridStep) line(0, y, this.w, y);
```

With:

```
stroke(240, 240, 240, 80);
for (let x = 0; x <= this.w; x += this.gridStep) line(x, 0, x, this.h);
for (let y = 0; y <= this.h; y += this.gridStep) line(0, y, this.w, y);
```

Lower opacity = softer spatial perception.

✨ 5. Slow Player Visual Presence

Update `draw()` in Player.js

Replace:

```
rect(this.x - 12, this.y - 12, 24, 24, 5);
```

With:

```
let pulse = sin(frameCount * 0.05) * 2;

rect(this.x - 12 - pulse, this.y - 12 - pulse, 24 + pulse*2, 24 + pulse*2, 6);
```

Now player “breathes”.

Not just moves.

🧠 What This Achieves (Conceptually)

You now have:

Before	After
Camera tracks	Camera drifts
Player moves	Player glides
Grid world	Atmospheric field
Input-driven	Emotion-driven

The world becomes:

> something you move through
> instead of
> something you control

✨ Final Experience Summary

This transforms your system from:

🎮 Navigation

into

🌌 Contemplation

Through:

- inertia
- delay
- softness
- subtle motion

Which fulfills the goal of:

✔ Scrolling world larger than screen
✔ Motion as emotional pacing
✔ Reflective camera presence

If you'd like, I can help you push this even further (e.g. parallax layers or slow colour transitions) for a stronger “meditative” aesthetic.

2nd Iteration:

which code am I replacing for step 3

2nd Response:

Great question 👍 — Step 3 is **not replacing anything**.

It’s something you **add**.

📍 Where Step 3 Goes

Open **sketch.js**

Find this section inside `draw()`:

```
// --- view state update (camera) ---
const maxCamX = max(0, level.w - width);
const maxCamY = max(0, level.h - height);

// Center camera on player, then clamp to legal camera range.
camX = constrain(player.x - width / 2, 0, maxCamX);
camY = constrain(player.y - height / 2, 0, maxCamY);
```

🛠️ First you already replaced this in Step 2 with:

```
let targetX = constrain(player.x - width / 2, 0, maxCamX);
let targetY = constrain(player.y - height / 2, 0, maxCamY);

camX = lerp(camX, targetX, 0.03);
camY = lerp(camY, targetY, 0.03);
```

✨ Step 3 = ADD this right below that

So your final camera section should look like:

```
// --- view state update (camera) ---
const maxCamX = max(0, level.w - width);
const maxCamY = max(0, level.h - height);

// soft follow
let targetX = constrain(player.x - width / 2, 0, maxCamX);
let targetY = constrain(player.y - height / 2, 0, maxCamY);

camX = lerp(camX, targetX, 0.03);
camY = lerp(camY, targetY, 0.03);

// 🌬️ breathing drift (ADD THIS PART)
let breathX = sin(frameCount * 0.01) * 8;
let breathY = cos(frameCount * 0.008) * 6;

camX += breathX;
camY += breathY;
```

🧘 Why this matters

Without this:

➡️ Camera just slowly follows

With this:

➡️ Camera *feels alive*, even when you stop moving

This is what creates the meditative “floating” experience.

If you'd like, I can now help you tune the breathing speed to feel more calm vs dreamy.

3rd Iteration:

i want something that evokes emotion, with more aesthetic

3rd Response:

Nice — I checked the piece.

What that example does well (emotionally) is:

👉 The **camera feels independent**
👉 The **world feels soft, not geometric**
👉 Movement feels like *floating through memory*, not navigating space

Right now your system is still:

> player-led

But theirs is:

> environment-led

So instead of just making motion smoother — we need to shift the *role of the camera*.

🎨 Emotional Difference

Your current build:

- camera = follows player
- player = centre of meaning

Their build:

- camera = observer
- player = just a presence inside atmosphere

This is what creates:

✨ introspection
✨ distance
✨ aesthetic calm

🧭 Design Direction Shift

To get closer to that:

We don’t just polish movement.

We:

1. Let the camera drift *on its own*

2. Let the player exist inside that drift

3. Make the world feel layered instead of flat

✨ Step 1 — Make Camera Semi-Autonomous

Right now camera always lerps to player.

Replace your camera logic in **sketch.js** with this:

```
let targetX = player.x - width / 2;
let targetY = player.y - height / 2;

// environmental drift
let envDriftX = sin(frameCount * 0.002) * 120;
let envDriftY = cos(frameCount * 0.0015) * 90;

targetX += envDriftX;
targetY += envDriftY;

targetX = constrain(targetX, 0, maxCamX);
targetY = constrain(targetY, 0, maxCamY);

// slower emotional lag
camX = lerp(camX, targetX, 0.015);
camY = lerp(camY, targetY, 0.015);
```

Now:

➡️ Camera no longer serves the player
➡️ Player exists inside a drifting world

This creates emotional distance.

✨ Step 2 — Add Depth (Parallax Layers)

Right now your world draws as one layer.

We fake depth by drawing soft background layers that move slower.

In **WorldLevel.js** → **drawWorld()**

Add BEFORE obstacles:

```
// far layer (moves slowest)
push();
translate(camX * 0.2, camY * 0.2);
noStroke();
fill(240, 245, 255, 120);
ellipse(this.w * 0.3, this.h * 0.3, 600, 400);
ellipse(this.w * 0.7, this.h * 0.6, 500, 350);
pop();

// mid layer
push();
translate(camX * 0.5, camY * 0.5);
fill(220, 235, 255, 90);
ellipse(this.w * 0.5, this.h * 0.4, 400, 300);
pop();
```

This mimics the aesthetic softness in the reference.

Now space feels deep instead of mapped.

✨ Step 3 — Remove “Game Feel”

Obstacles currently look like UI blocks.

Replace obstacle fill:

```
fill(170, 190, 210);
rect(o.x, o.y, o.w, o.h, o.r ?? 0);
```

With:

```
push();
translate(o.x + o.w/2, o.y + o.h/2);

let wobble = sin(frameCount * 0.01 + o.x) * 3;

fill(200, 220, 255, 120);
ellipse(0, 0, o.w + wobble, o.h + wobble);

pop();
```

Now they feel like:

☁️ memories
🌫️ clouds
🫧 emotional landmarks

Instead of walls.

✨ Step 4 — Fade Edges of the Screen

This is key to aesthetic mood.

In **sketch.js** → **after pop()**, add:

```
// soft vignette
noStroke();
for (let i = 0; i < 120; i++) {
  fill(255, 255, 255, i * 0.5);
  rect(i, i, width - i*2, height - i*2);
}
```

This creates:

🎞️ dream-like framing

✨ Step 5 — Slow Player Further

Update your player velocity lerp:

this.vx = lerp(this.vx ?? 0, ax * this.s, 0.02);
this.vy = lerp(this.vy ?? 0, ay * this.s, 0.02);

Lower response = more meditative.

🌌 Result

Now the experience becomes:

Instead of:

> moving through a world

You are:

> witnessing a drifting landscape
> where movement reveals emotion

Closer to your reference piece.

🧠 UX Framing (for your write-up)

You can describe this as:

> A camera that does not follow, but wanders — allowing the player to inhabit rather than control space.

If you'd like, I can push this even more toward:

- memory-like colour shifts
- time-of-day transitions
- ambient particle motion

Which would elevate the aesthetic further.

4th Iteration:

[insert current code files] I see no player and no movement allowed, something is off

4th Response:

Yes — I can see exactly what broke 👍
You didn't lose the player by accident — your **WorldLevel.js has a logic error** that stops the world from drawing properly.

This line is the problem:

Inside `drawWorld()` you added:

```
push();
translate(o.x + o.w / 2, o.y + o.h / 2);
```

But ❗ **there is no loop anymore**

So:

👉 `o` is undefined
👉 drawWorld crashes silently
👉 player never appears
👉 movement feels frozen

That's why you see nothing.

🔴 What Broke

Your current obstacles section looks like this

```
// Obstacles
noStroke();
push();
translate(o.x + o.w / 2, o.y + o.h / 2);

let wobble = sin(frameCount * 0.01 + o.x) * 3;

fill(200, 220, 255, 120);
ellipse(0, 0, o.w + wobble, o.h + wobble);

pop();
```

But originally it had:

```
for (const o of this.obstacles)
```

Without that loop → `o` does not exist.

So nothing renders after this point.

✅ FIX — Restore Loop

Replace your entire **Obstacles section** with this:

```
// Obstacles (soft floating forms)
```

```
noStroke();
for (const o of this.obstacles) {
  push();
  translate(o.x + o.w / 2, o.y + o.h / 2);

  let wobble = sin(frameCount * 0.01 + o.x) * 3;

  fill(200, 220, 255, 120);
  ellipse(0, 0, o.w + wobble, o.h + wobble);

  pop();
}
```

Now:

✔ o exists
✔ world draws
✔ player appears
✔ motion returns

🟡 Second Issue (Parallax using camX)

You also added this:

```
translate(camX * 0.2, camY * 0.2);
```

But **WorldLevel.js doesn't own camX / camY**

So it can sometimes behave unpredictably.

For now — leave it, since it's working visually.

Later we can pass camera in properly if needed.

🟢 After Fix You Should See

- Player breathing square
- Slow drifting camera
- Floating blobs instead of hard blocks
- Movement works again

🧠 Why This Happened (Good Learning Moment)

You moved from:

👉 deterministic drawing
to
👉 expressive drawing

And expressive drawing often breaks structure if loops disappear.

So always check:

> "Am I still iterating through world data?"

Apply the fix and reload — your player should be back.

If it still feels “stuck”, I can tune motion next to match the emotional pacing of your reference piece more closely.

5th Iteration:

[insert current screen] There is a big rectangle blurb blocking player and this is not what I want, stick with requirement please. Create a reflective or meditative camera

experience that scrolls through a world larger than the screen. Use pacing and motion to evoke emotion. Hide small interactive symbols or objects for the camera to “discover.

5th Response:

Perfect — this screenshot helped a lot.

That **big white rectangle blur** is coming from this vignette code in your `sketch.js`:

```
for (let i = 0; i < 120; i++) {
  fill(255, 255, 255, i * 0.5);
  rect(i, i, width - i * 2, height - i * 2);
}
```

Right now it’s not a vignette — it’s a **white box stack**.

So it's literally covering your world and player.

This breaks both:

❌ visibility
❌ meditative exploration

And you're right — this doesn't meet the requirement.

🎯 Let’s Reset Toward the Assignment Goal

The requirement is:

> Reflective camera
> World larger than screen
> Pacing evokes emotion
> Hidden discoverable objects

So we want:

✔ slow scroll
✔ quiet motion
✔ subtle discoveries
❌ visual obstruction

✅ Step 1 — REMOVE the White Block

Delete this entirely from `sketch.js`:

```
for (let i = 0; i < 120; i++) {
  fill(255, 255, 255, i * 0.5);
  rect(i, i, width - i * 2, height - i * 2);
}
```

✨ Step 2 — Replace With a Soft Shadow Frame (Meditative)

Add this instead:

```
// soft edge fade
noStroke();
for (let i = 0; i < 80; i++) {
  fill(0, 0, 0, i * 0.8);
  rect(i, 0, 1, height);
  rect(width - i, 0, 1, height);
  rect(0, i, width, 1);
  rect(0, height - i, width, 1);
}
```

Now:

➡️ edges darken
➡️ centre remains visible
➡️ camera feels framed

Like memory.

✨ Step 3 — Add Hidden Discoverables

We now introduce **quiet symbols**.

Open `WorldLevel.js`

Inside constructor add:

```
this.symbols = [];

for (let i = 0; i < 20; i++) {
  this.symbols.push({
    x: random(this.w),
    y: random(this.h),
    size: random(8, 16)
  });
}
```

Then inside `drawWorld()` at the bottom add:

```
// hidden symbols
for (const s of this.symbols) {
```

```
  let distToCam = dist(s.x, s.y, camX + width/2, camY + height/2);

  let alpha = map(distToCam, 0, 400, 200, 0);
  alpha = constrain(alpha, 0, 200);

  noStroke();
  fill(120, 160, 255, alpha);

  ellipse(s.x, s.y, s.size);
}
```

Now:

✨ Symbols appear only when you approach
✨ Camera “discovers” them
✨ World feels intentional

✨ Step 4 — Make Camera More Meditative

In `sketch.js` change lerp speed:

```
camX = lerp(camX, targetX, 0.01);
camY = lerp(camY, targetY, 0.01);
```

Slower = more reflective pacing.

🧘 Result

You now have:

- A scrolling world larger than screen ✔
- Camera-led discovery ✔
- Hidden quiet objects ✔
- Motion that evokes curiosity ✔
- No visual obstruction ✔

Instead of:

> looking at the world

You are:

> uncovering it slowly

If you'd like, we can now make symbols react (pulse / fade / shimmer) to deepen emotional tone.

6th Iteration:

Is good now, but I want to make it aesthetically pleasing with good ui

6th Response:

Nice — now that the *experience works*, we can elevate the **aesthetic + UI** without breaking the meditative intent.

Right now your system has:

✔ reflective motion
✔ discovery
✔ quiet pacing

But visually it still feels like:

> a prototype

We want it to feel like:

> a designed experience

🎨 Design Direction

Meditative UI should be:

- minimal
- soft
- intentional
- non-instructional

So instead of “HUD info”, we move toward:

✨ atmospheric UI
✨ emotional guidance
✨ invisible interaction

✨ Step 1 — Remove Technical HUD

Your current top-left debug text breaks immersion

In `draw()` remove:

level.drawHUD(player, camX, camY);

Now we design a **presence-based UI** instead.

✨ Step 2 — Add Ambient Background Gradient

In `drawBackground()` replace:

background(220);

with:

```
for (let i = 0; i < height; i++) {
  let inter = map(i, 0, height, 0, 1);
  let c = lerpColor(
    color(245, 248, 255),
    color(225, 235, 250),
    inter
  );
  stroke(c);
  line(0, i, width, i);
}
```

Now the world feels like atmosphere, not canvas.

✨ Step 3 — Soft Player Presence

Update Player `draw()`:

Replace fill:

fill(50, 110, 255);

With:

```
let glow = sin(frameCount * 0.04) * 40 + 120;
fill(160, 190, 255, glow);
```

Now player feels like a presence, not an object.

✨ Step 4 — Symbol UI Glow

Inside your hidden symbols loop, add:

let pulse = sin(frameCount * 0.05 + s.x) * 2;

ellipse(s.x, s.y, s.size + pulse);

And add a halo:

fill(120, 160, 255, alpha * 0.2);
ellipse(s.x, s.y, s.size * 3);

Now discoveries feel meaningful.

✨ Step 5 — Minimal UI Cue

Instead of instructions, add a subtle guiding line.

At bottom of `draw()`:

stroke(200, 210, 240, 60);
line(width/2 - 40, height - 30, width/2 + 40, height - 30);

This becomes:

➡️ a visual anchor
➡️ not text-based direction

✨ Step 6 — Add Motion Calm

Slow world slightly:

In camera lerp:

```
camX = lerp(camX, targetX, 0.008);
camY = lerp(camY, targetY, 0.008);
```

This improves emotional pacing.

🧘 Resulting UI Philosophy

Instead of telling the user:

> “Explore”

The UI now *feels like*:

> The world is waiting to be discovered

✨ What This Achieves

Before	After
Debug overlay	Ambient presence
Sharp player	Soft entity
Static symbols	Living signals

Blank space	Atmospheric depth

Now your piece reads as:

> a reflective camera journey

—not a game.

If you'd like, I can help you add a slow colour shift over time (sunrise → dusk mood) to push emotional tone even further.

7th Iteration:

[insert image screen] I want to change my player to be a sparking circle, and change the rectangular border to be a prettier environment

7th Response:

Perfect — this is a great direction.

Right now:

✔ your experience works
✔ your symbols feel subtle

But visually:

❌ the player still feels like a UI object
❌ the border feels like a frame, not an environment

Let’s transform:

➡️ Player → ✨ sparkling presence
➡️ Border → 🌫️ atmospheric world edges

✨ PART 1 — Make Player a Sparkling Circle

Open **Player.js**

Replace your entire `draw()` with this:

```
draw() {
  noStroke();

  // breathing glow
  let glow = sin(frameCount * 0.05) * 20 + 60;

  // outer aura
  fill(180, 210, 255, 40);
  ellipse(this.x, this.y, glow * 2);

  // mid aura
  fill(160, 200, 255, 80);
  ellipse(this.x, this.y, glow);

  // core
  fill(140, 180, 255, 180);
  ellipse(this.x, this.y, 14);

  // sparkle particles
  for (let i = 0; i < 3; i++) {
    let angle = frameCount * 0.03 + i * TWO_PI/3;
    let r = 18 + sin(frameCount * 0.05 + i) * 4;

    let sx = this.x + cos(angle) * r;
    let sy = this.y + sin(angle) * r;
```

```
    fill(200, 220, 255, 180);
    ellipse(sx, sy, 3);
  }
}
```

Now the player becomes:

✨ alive
✨ soft
✨ emotional

Not mechanical.

🌫️ PART 2 — Replace Rectangle Border with Atmosphere

Your current border is coming from this in `sketch.js`:

```
for (let i = 0; i < 80; i++) {
  fill(0, 0, 0, i * 0.8);
  rect(i, 0, 1, height);
  rect(width - i, 0, 1, height);
  rect(0, i, width, 1);
  rect(0, height - i, width, 1);
}
```

Delete this.

✨ Replace with Floating Fog

Add this instead at the end of `draw()`:

```
// soft atmospheric edges
noStroke();

for (let i = 0; i < 150; i++) {

  let alpha = map(i, 0, 150, 0, 60);

  fill(230, 240, 255, alpha);

  ellipse(-50, height/2, 200 + i*2, height + 200);
  ellipse(width+50, height/2, 200 + i*2, height + 200);

  ellipse(width/2, -50, width + 200, 200 + i*2);
  ellipse(width/2, height+50, width + 200, 200 + i*2);
}
```

This creates:

🌫️ drifting fog edges
instead of
🧱 hard frame

So now the world feels infinite — not boxed.

✨ OPTIONAL — Add Slow Fog Motion

To make it even more alive:

Replace the ellipse positions with:

let drift = sin(frameCount * 0.01) * 20;

Then:

ellipse(-50 + drift, height/2, 200 + i*2, height + 200);
ellipse(width+50 - drift, height/2, 200 + i*2, height + 200);
ellipse(width/2, -50 + drift, width + 200, 200 + i*2);
ellipse(width/2, height+50 - drift, width + 200, 200 + i*2);

Now the environment breathes.

🌌 Result

You now have:

✔ Sparkling emotional player
✔ World that fades into atmosphere
✔ No rigid UI border
✔ Meditative spatial feeling

Instead of:

> framed space

You get:

> dissolving space

If you'd like, we can next make the hidden symbols shimmer when discovered for stronger emotional reward.