Process & Decision Documentation

Project/Assignment Decisions

Inspired by the concept of a scrolling world larger than the screen, I designed this project as a reflective camera experience that emphasizes pacing, motion, and spatial scale rather than goal-oriented gameplay. The project directly aligns with the assignment's focus on world space versus screen space, camera transformation using translate(), and structured state updates within the draw loop.

The world exists in WORLD coordinates that exceed the canvas dimensions, while the camera maintains its own WORLD position and renders the scene by translating the view relative to that camera. Player movement is governed by velocity, acceleration, and friction rather than direct position changes, ensuring motion feels gradual and weighted. The camera follows the player using interpolation (lerp) instead of snapping directly to the target, creating a subtle delay that contributes to the meditative pacing.

This separation between player state, camera state, and rendering logic ensures that spatial relationships are clearly defined and that the experience is driven by motion dynamics rather than static positioning. The design prioritizes smoothness, inertia, and scale as emotional tools.

GenAI Documentation

GenAI was used to support structured documentation writing and refinement of technical descriptions. All core gameplay logic, movement systems, camera behavior, and visual structure were implemented manually.

Summary of Process (Human + Tool)

• Defined the world as a coordinate space larger than the canvas to establish scrolling as a structural requirement.

• Separated WORLD coordinates (player, features) from SCREEN coordinates (HUD).

• Implemented camera positioning as a WORLD-space top-left reference translated via translate(-cam.x, -cam.y).

• Replaced direct positional movement with acceleration, velocity, and friction to create inertia-based motion.

• Introduced camera easing using interpolation to produce delayed follow behavior.

• Iteratively tuned constants (ACCEL, FRICTION, CAM_EASE, MAX_SPEED) to adjust emotional pacing.

• Developed a procedurally generated background to provide subtle spatial reference without overwhelming the scene.

• Tested movement across extreme world edges to ensure consistent camera behavior without clamping errors.

• Refined motion parameters to balance responsiveness with calmness.

Decision Points & Trade-offs

• Chose a physics-inspired movement system (velocity + friction) instead of direct key-based translation to evoke weight and smoothness.
• Used interpolation for camera tracking, trading precision for emotional delay and softness.
• Avoided win conditions or objectives to maintain a contemplative focus.
• Kept the visual design minimal, prioritizing motion and scale over detailed environmental assets.
• Did not constrain the camera to world bounds, allowing exploration to emphasize the world's scale rather than containment.
• Used deterministic procedural placement for background elements instead of fully randomized regeneration each frame to maintain stability.

Verification & Judgement

• Verified that camera translation correctly reflects WORLD-to-SCREEN transformation by observing relative object motion.
• Confirmed that easing behavior produces visible lag without breaking spatial alignment.

• Tested diagonal movement normalization to ensure consistent velocity magnitude.
• Checked edge constraints to prevent the player from exiting WORLD bounds.
• Evaluated whether pacing adjustments meaningfully changed the emotional tone of movement.
• Assessed whether the experience communicates scale and stillness without additional mechanics.

Limitations, Dead Ends, or Open Questions

• The project currently relies solely on motion dynamics for emotional impact; additional environmental layers (parallax, lighting, sound) could deepen the experience.
• The camera system does not yet incorporate zoom or rotation for further cinematic variation.
• The world is procedurally styled but not narratively structured, limiting interpretive context.
• There is no adaptive pacing system; emotional tone is constant rather than responsive.
• Future iterations could experiment with dynamic camera framing, subtle zoom breathing, or ambient sound design to enhance immersion.

Appendix

Challenge: Create a reflective or meditative camera experience

that scrolls through a world larger than the screen.

Use pacing and motion to evoke emotion.

Language: p5.js

Attached is an example consists of component files to use as a foundation

Got you — we'll build this **incrementally**, using *your exact foundation* (world coords → cam.x/cam.y → translate(-cam.x, -cam.y) → HUD after pop()).

Below is **Step 1 only**: make the world feel "bigger-than-screen" *and* start shifting the vibe from "grid demo" to "meditative space," without changing the camera logic yet.

Step 1 — Bigger world + calmer world visuals (no camera changes)

What stays the same

- Player moves in **WORLD** space
- Camera stays **centered** on player (your current code)
- World drawn inside push(); translate(-cam.x, -cam.y); ... pop();

What we change

1. Make the world larger
2. Replace the harsh grid / bright background with:

- a **soft vertical gradient**
- a few **quiet "stone" shapes** (procedural, stable)

Edit your sketch.js with these small changes

1) Change the world size

Replace:

```
const WORLD_W = 2400;

const WORLD_H = 1600;
```

with:

```
const WORLD_W = 5200;

const WORLD_H = 3400;
```

2) Add a helper function (paste near the bottom of the file)

Add this **new function** anywhere outside draw() (usually below draw() is fine):

```
function drawMeditativeBackground() {

 // soft sky/ground gradient across the WORLD

 noStroke();

 for (let y = 0; y < WORLD_H; y += 18) {

  const n = y / WORLD_H;
```

```
    fill(
      14 + 18 * (1 - n),
      16 + 18 * (1 - n),
      22 + 30 * (1 - n)
    );
    rect(0, y, WORLD_W, 18);
  }

  // sparse "stones" to create gentle points of reference
  for (let i = 0; i < 140; i++) {
    const x = (i * 137) % WORLD_W; // deterministic placement
    const y = (i * 251) % WORLD_H;
    const r = 10 + (i % 7) * 3;

    fill(180, 190, 210, 35);
    ellipse(x, y, r * 1.8, r);

    fill(220, 230, 255, 16);
    ellipse(x - r * 0.2, y - r * 0.15, r * 0.6, r * 0.35);
  }
}
```

3) Use it inside your WORLD drawing block

In draw(), inside the push() / translate(-cam.x, -cam.y) section, replace your current world background + grid lines:

Replace this whole part:

```
// World background rectangle (so you can see the "world area")
```

```
noStroke();
fill(235);
rect(0, 0, WORLD_W, WORLD_H);

// Grid lines make camera motion easy to see
stroke(245);
for (let x = 0; x <= WORLD_W; x += 160) line(x, 0, x, WORLD_H);
for (let y = 0; y <= WORLD_H; y += 160) line(0, y, WORLD_W, y);
```

With this:

```
drawMeditativeBackground();
```

That's it for Step 1.

What you should notice now

- You're clearly moving through a **much larger** space than the canvas.
- The scene stops feeling like a technical demo and starts feeling like a **place**.
- Motion is still "gamey" (camera snaps), but we haven't touched pacing yet.

Next steps (just preview, not doing them yet)

- **Step 2:** Make the camera *lag softly* behind the player (lerp the camera) → instant calm.
- **Step 3:** Smooth player acceleration/deceleration (less "grid movement").
- **Step 4:** Add slow drifting motes + parallax.
- **Step 5:** Add subtle "breathing" zoom.

When you're ready, say **"Step 2"** and I'll give you the *smallest possible* change to get that smooth meditative camera.

Perfect — **Step 2 = soften the camera** (same logic, just add *lag/easing*). This is the single biggest "meditative" upgrade.

Your current foundation sets camera directly from player each frame. We'll change it so the camera **eases toward** the player using lerp(). This keeps everything else identical: same world drawing block, same translate(-cam.x, -cam.y).

Step 2 — Camera lag (easing)

What you change

In draw(), you likely have something like:

```
cam.x = player.x - width / 2;

cam.y = player.y - height / 2;
```

Replace it with:

```
// Desired camera position (same as before)

const targetCamX = player.x - width / 2;

const targetCamY = player.y - height / 2;


// Ease toward target (lower = slower, more "floaty")

const CAM_EASE = 0.06;


cam.x = lerp(cam.x, targetCamX, CAM_EASE);

cam.y = lerp(cam.y, targetCamY, CAM_EASE);
```

That's it.

Tiny optional add-on (still Step 2, still minimal)

To prevent the camera from drifting outside the world edges, clamp it **after** easing:

```
cam.x = constrain(cam.x, 0, WORLD_W - width);

cam.y = constrain(cam.y, 0, WORLD_H - height);
```

(Use this if you notice empty space outside your world.)

What you should feel now

- The player moves *first*, the camera follows *a beat later*.
- Motion becomes **gentler**, less “locked to the player.”
- The world feels heavier, calmer, more cinematic.

Say **Step 3** when you’re ready: we’ll smooth the *player’s* movement (acceleration + friction), so it stops feeling like a sprite sliding on a grid.

Step 3 — **smooth the player motion** (acceleration + friction), so it stops feeling “digital” and starts feeling *weighty / breathy*.

We’ll keep the same structure you already have (player in WORLD space, camera easing, world drawn with translate(-cam.x, -cam.y)).

Step 3A — Add velocity to the player

1) In your player object, add vx and vy

Find where you define your player (often near the top), and add:

```
player.vx = 0;

player.vy = 0;
```

If your player is created like:

```
let player = { x: 300, y: 300, w: 40, h: 40, speed: 5 };
```

Change it to:

```
let player = { x: 300, y: 300, w: 40, h: 40, vx: 0, vy: 0 };
```

Step 3B — Replace “direct position movement” with acceleration + friction

2) In draw(), find your movement code

You probably have something like:

```
if (keyIsDown(LEFT_ARROW)) player.x -= 5;

if (keyIsDown(RIGHT_ARROW)) player.x += 5;

if (keyIsDown(UP_ARROW)) player.y -= 5;
if (keyIsDown(DOWN_ARROW)) player.y += 5;
```

Replace that whole block with this:

```
// --- Player motion tuning (meditative pacing) ---
const ACCEL = 0.35;     // how quickly you speed up
const FRICTION = 0.90;  // how quickly you slow down (closer to 1 = floatier)
const MAX_SPEED = 4.2;  // cap speed so it stays calm

let ax = 0;
let ay = 0;

if (keyIsDown(LEFT_ARROW) || keyIsDown(65)) ax -= ACCEL;   // A
if (keyIsDown(RIGHT_ARROW) || keyIsDown(68)) ax += ACCEL;  // D
if (keyIsDown(UP_ARROW) || keyIsDown(87)) ay -= ACCEL;     // W
if (keyIsDown(DOWN_ARROW) || keyIsDown(83)) ay += ACCEL;   // S

// Apply acceleration
player.vx += ax;
player.vy += ay;

// Apply friction (ease to stillness)
player.vx *= FRICTION;
player.vy *= FRICTION;

// Cap speed (keeps it dreamy, not twitchy)
player.vx = constrain(player.vx, -MAX_SPEED, MAX_SPEED);
player.vy = constrain(player.vy, -MAX_SPEED, MAX_SPEED);
```

```
// Move in world space

player.x += player.vx;

player.y += player.vy;
```

Step 3C — Keep player inside the world (soft clamp)

Right after player.x += player.vx / player.y += player.vy, add:

```
player.x = constrain(player.x, 0, WORLD_W);

player.y = constrain(player.y, 0, WORLD_H);
```

(If you use a player radius/size, clamp using that instead.)

What you should feel now

- Movement eases in and eases out (no more instant stop/start).
- Combined with Step 2 camera easing, the whole thing becomes **floaty and meditative**.

If it feels too slippery or too snappy

- **More calm / floaty:** FRICTION = 0.94, MAX_SPEED = 3.6
- **More responsive:** FRICTION = 0.86, ACCEL = 0.45

Say **Step 4** and we'll add the first "emotion layer": **slow drifting motes + subtle parallax** (still minimal changes).