Process & Decision Documentation

Project/Assignment Decisions

Create a reflective or meditative camera experience that scrolls through a world larger than the screen.

GenAI Documentation

Human Decision Point(s)

- I chose a galaxy theme as my world
- I decided to add some hidden some stars and planets to enhance the galaxy theme
- I then changed the sizes of the planets by myself because they were too big.

Transcript

modify each of these codes to create a reflective or meditative camera experience that scrolls through a world larger than the screen. Use pacing and motion to evoke emotion. I want a galaxy theme world. Hide small interactive stars for the camera to “discover. Player code: class Player { constructor(x, y, speed) { this.x = x; this.y = y; this.s = speed ?? 3; } updateInput() { const dx = (keyIsDown(RIGHT_ARROW) || keyIsDown(68)) - (keyIsDown(LEFT_ARROW) || keyIsDown(65)); const dy = (keyIsDown(DOWN_ARROW) || keyIsDown(83)) - (keyIsDown(UP_ARROW) || keyIsDown(87)); const len = max(1, abs(dx) + abs(dy)); this.x += (dx / len) * this.s; this.y += (dy / len) * this.s; } draw() { fill(50, 110, 255); noStroke(); rect(this.x - 12, this.y - 12, 24, 24, 5); } } Sketch code: const VIEW_W = 800; const VIEW_H = 480; let worldData; let level; let player; let camX = 0; let camY = 0; function preload() { worldData = loadJSON("world.json"); // load JSON before setup [web:122] } function setup() { createCanvas(VIEW_W, VIEW_H); textFont("sans-serif"); textSize(14); level = new WorldLevel(worldData); const start = worldData.playerStart ?? { x: 300, y: 300, speed: 3 }; player = new Player(start.x, start.y, start.speed); camX = player.x - width / 2; camY = player.y - height / 2; } function draw() { player.updateInput(); // Keep player inside world player.x = constrain(player.x, 0, level.w); player.y = constrain(player.y, 0, level.h); // Target camera (center on player) let targetX = player.x - width / 2; let targetY = player.y - height / 2; // Clamp target camera safely const maxCamX = max(0, level.w - width); const maxCamY = max(0, level.h - height); targetX = constrain(targetX, 0, maxCamX); targetY = constrain(targetY, 0, maxCamY); // Smooth follow using the JSON knob const camLerp = level.camLerp; // ← data-driven now camX = lerp(camX, targetX, camLerp); camY = lerp(camY, targetY, camLerp); level.drawBackground(); push(); translate(-camX, -camY); level.drawWorld(); player.draw(); pop(); level.drawHUD(player, camX, camY); } 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); } } World level code: class WorldLevel { constructor(json)

{ this.schemaVersion = json.schemaVersion ?? 1; this.w = json.world?.w ?? 2400; this.h = json.world?.h ?? 1600; this.bg = json.world?.bg ?? [235, 235, 235]; this.gridStep = json.world?.gridStep ?? 160; this.obstacles = json.obstacles ?? []; // NEW: camera tuning knob from JSON (data-driven) this.camLerp = json.camera?.lerp ?? 0.12; } drawBackground() { background(220); } drawWorld() { noStroke(); fill(this.bg[0], this.bg[1], this.bg[2]); rect(0, 0, this.w, this.h); 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); noStroke(); fill(170, 190, 210); for (const o of this.obstacles) rect(o.x, o.y, o.w, o.h, o.r ?? 0); } drawHUD(player, camX, camY) { noStroke(); fill(20); text("Example 4 — JSON world + smooth camera (lerp).", 12, 20); text("camLerp(JSON): " + this.camLerp + " Player: " + (player.x | 0) + "," + (player.y | 0) + " Cam: " + (camX | 0) + "," + (camY | 0), 12, 40,); } }

Love this direction. We'll transform your world into a **slow, drifting galaxy** where movement feels intentional and emotional. The camera will glide gently, stars will shimmer subtly, and hidden stars will glow when discovered.

Below are **modified versions** of your three files.

🌌 1. Modified Player Code (Ethereal Drift Movement)

We'll:

- Slow the base speed
- Add slight easing (floaty motion)
- Replace the square with a glowing orb

```
class Player {
  constructor(x, y, speed) {
    this.x = x;
    this.y = y;
    this.s = speed ?? 1.8; // slower, meditative pace
    this.vx = 0;
    this.vy = 0;
  }

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

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

    // gentle acceleration
    this.vx += dx * 0.15;
    this.vy += dy * 0.15;

    // soft damping (floating in space)
    this.vx *= 0.92;
    this.vy *= 0.92;

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

  draw() {
    noStroke();

    // soft glow
    for (let i = 40; i > 0; i -= 8) {
      fill(120, 180, 255, 20);
      ellipse(this.x, this.y, i);
    }

    // core
    fill(180, 220, 255);
    ellipse(this.x, this.y, 14);
  }
}
```

This now feels like drifting consciousness instead of a block moving.

🌌 2. Modified Sketch Code (Slow Emotional Camera)

We’ll:

Make camera slower and more cinematic

Add subtle camera breathing motion

Dark space background

```
const VIEW_W = 800;
const VIEW_H = 480;

let worldData;
let level;
let player;

let camX = 0;
let camY = 0;

function preload() {
  worldData = loadJSON("world.json");
}

function setup() {
  createCanvas(VIEW_W, VIEW_H);
  level = new WorldLevel(worldData);

  const start = worldData.playerStart ?? { x: 1200, y: 800,
speed: 1.8 };
  player = new Player(start.x, start.y, start.speed);

  camX = player.x - width / 2;
  camY = player.y - height / 2;
}

function draw() {
  player.updateInput();

  player.x = constrain(player.x, 0, level.w);
  player.y = constrain(player.y, 0, level.h);

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

  const maxCamX = max(0, level.w - width);
  const maxCamY = max(0, level.h - height);

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

  // slower, more reflective follow
  camX = lerp(camX, targetX, 0.04);
  camY = lerp(camY, targetY, 0.04);

  level.drawBackground();

  push();
  translate(-camX, -camY);
  level.drawWorld(player);
  player.draw();
  pop();
}
```

Camera lerp changed from `0.12` → `0.04`
This makes it **float behind the player** emotionally.

🌌 3. Modified WorldLevel Code (Galaxy + Hidden Discoverable Stars)

We will:

- Remove grid
- Create layered stars (parallax)
- Add hidden interactive stars
- Glow when discovered

```
class WorldLevel {
  constructor(json) {
    this.w = json.world?.w ?? 4000;
    this.h = json.world?.h ?? 3000;

    this.camLerp = json.camera?.lerp ?? 0.04;

    this.stars = [];
    this.hiddenStars = [];

    // generate background stars
    for (let i = 0; i < 600; i++) {
      this.stars.push({
        x: random(this.w),
        y: random(this.h),
        size: random(1, 3),
        alpha: random(100, 255)
      });
    }

    // hidden discoverable stars
    for (let i = 0; i < 25; i++) {
      this.hiddenStars.push({
        x: random(this.w),
        y: random(this.h),
        found: false
      });
    }
  }

  drawBackground() {
    // deep galaxy gradient
    for (let y = 0; y < height; y++) {
      let inter = map(y, 0, height, 0, 1);
      let c = lerpColor(
        color(10, 10, 35),
        color(2, 0, 15),
        inter
      );
      stroke(c);
      line(0, y, width, y);
    }
  }
  drawWorld(player) {
    noStroke();

    // draw distant stars
    for (const s of this.stars) {
      fill(255, 255, 255, s.alpha + sin(frameCount * 0.02 + s.x) * 40);
      ellipse(s.x, s.y, s.size);
    }

    // hidden stars interaction
    for (const hs of this.hiddenStars) {
      const d = dist(player.x, player.y, hs.x, hs.y);

      if (d < 60 && !hs.found) {
        hs.found = true;
      }

      if (hs.found) {
        for (let i = 30; i > 0; i -= 6) {
          fill(255, 240, 180, 15);
          ellipse(hs.x, hs.y, i);
        }
        fill(255, 240, 180);
        ellipse(hs.x, hs.y, 6);
      }
    }
  }

  drawHUD(player, camX, camY) {
    fill(255, 150);
    textSize(12);
    textAlign(LEFT);
    text("Drift. Discover. Breathe.", 20, height - 20);
  }
}
```

🌟 What This Creates Emotionally

- Slower movement → introspection
- Camera lag → feeling of memory or echo
- Large world (4000x3000) → isolation
- Hidden stars → reward through exploration
- Soft glows → warmth in darkness
- No grid → infinite space feeling

🌌 Optional Enhancement (If You Want It Even More Cinematic)

Add:

- Soft ambient music
- Subtle particle dust trails behind player
- Slight zoom in/out breathing effect
- Very faint nebula clouds (low opacity shapes)

If you want, I can now:

- Add parallax galaxy layers
- Add a soft zoom breathing camera
- Add constellation connections when multiple stars are found
- Or convert this into a full poetic interactive experience 🌙

can you add some planets

Yes. Let's add **slow, majestic planets** drifting in the galaxy — large, atmospheric, layered with rings and glow — so the camera discovers them gradually as it scrolls.

We'll modify only the `WorldLevel` class.

This will add:

Large planets placed across the world

Subtle rotation effect

Soft glow halo

Occasional ringed planets

Gentle parallax illusion through slow shading movement

🌌 Updated `WorldLevel` with Planets

Replace your current `WorldLevel` class with this:

```
class WorldLevel {
  constructor(json) {
    this.w = json.world?.w ?? 4000;
    this.h = json.world?.h ?? 3000;

    this.camLerp = json.camera?.lerp ?? 0.04;

    this.stars = [];
    this.hiddenStars = [];
    this.planets = [];

    // ⭐ Background stars
    for (let i = 0; i < 600; i++) {
      this.stars.push({
        x: random(this.w),
        y: random(this.h),
        size: random(1, 3),
        alpha: random(100, 255)
      });
    }

    // 🌟 Hidden discoverable stars
    for (let i = 0; i < 25; i++) {
```

```
    this.hiddenStars.push({
      x: random(this.w),
      y: random(this.h),
      found: false
    });
  }

  // 🪐 Planets
  for (let i = 0; i < 8; i++) {
    this.planets.push({
      x: random(400, this.w - 400),
      y: random(400, this.h - 400),
      size: random(120, 260),
      color: color(
        random(80, 200),
        random(60, 150),
        random(150, 255)
      ),
      hasRings: random() > 0.5,
      rotationOffset: random(1000)
    });
  }
}

drawBackground() {
  for (let y = 0; y < height; y++) {
    let inter = map(y, 0, height, 0, 1);
    let c = lerpColor(
      color(10, 10, 35),
      color(2, 0, 15),
      inter
    );
    stroke(c);
    line(0, y, width, y);
  }
}

drawWorld(player) {
  noStroke();
```

```
    // ✨ Twinkling stars
    for (const s of this.stars) {
      fill(255, 255, 255, s.alpha + sin(frameCount * 0.02 + s.x) * 40);
      ellipse(s.x, s.y, s.size);
    }

    // 🪐 Planets (draw BEFORE hidden stars so they feel distant)
    for (const p of this.planets) {

      push();
      translate(p.x, p.y);

      // glow halo
      for (let i = p.size * 1.4; i > p.size; i -= 15) {
        fill(red(p.color), green(p.color), blue(p.color), 8);
        ellipse(0, 0, i);
      }

      // planet body
      fill(p.color);
      ellipse(0, 0, p.size);

      // subtle animated shading
      fill(0, 40);
      ellipse(
        sin(frameCount * 0.002 + p.rotationOffset) * 10,
        0,
        p.size * 0.9
      );

      // rings
      if (p.hasRings) {
        stroke(255, 120);
        noFill();
        strokeWeight(2);
        ellipse(0, 0, p.size * 1.6, p.size * 0.5);
      }

      pop();
    }

    // 🌟 Hidden discoverable stars
    for (const hs of this.hiddenStars) {
      const d = dist(player.x, player.y, hs.x, hs.y);

      if (d < 60 && !hs.found) {
        hs.found = true;
      }

      if (hs.found) {
        for (let i = 30; i > 0; i -= 6) {
          fill(255, 240, 180, 15);
          ellipse(hs.x, hs.y, i);
        }
        fill(255, 240, 180);
        ellipse(hs.x, hs.y, 6);
      }
    }
  }

  drawHUD() {
    fill(255, 150);
    textSize(12);
    text("Drift. Discover. Orbit.", 20, height - 20);
  }
}
```

🌠 What This Adds Emotionally

🪐 Large slow planets create:

Scale

Silence

A feeling of cosmic loneliness

✨ Rings add elegance and contrast
🌌 Soft glow halos make them feel sacred
🎥 Slow camera + large planets = cinematic reveal moments

If you want next-level polish, I can add:

- Planets that subtly react when approached
- Orbiting moons
- Nebula clouds
- A giant distant sun
- Parallax depth layers
- Constellations forming when stars are discovered

Tell me how cinematic you want this to feel.