Process & Decision Documentation

Anushka Kshirsagar

Project/Assignment Decisions

Focus on:

- Making a calm dark silent night themed screen for calmness and breathing exercise that focuses on users to discover hidden shiny objects to get an calm positive message
- Remove the player entirely and make the camera the main subject of the experience
- Add hidden glowing symbol objects scattered across a wide dreamscape world that the user can click to discover
- Create a calming breathing exercise overlay that appears when a symbol/object is clicked. Adjust pacing, motion, and visual atmosphere to evoke a calm and reflective emotion

Entry Header

Name: Anushka Kshirsagar

Primary responsibility for this work: Creative, prompting and editing

Goal of Work Session

Briefly describe what you were trying to accomplish during this phase of the assignment.

Tools, Resources, or Inputs Used

- GenAI tools (Claude.ai)
- Prior code from LEARN under the “Week5” examples
- 302 Week 5; Part 2 lecture notes (Camera2D, WorldLevel, levels.json)

GenAI Documentation

Date Used: February 17 and 18th 2026

Tool Disclosure: Claude.ai

Purpose of Use: Giving me code to translate my creative concept for calmness into working p5.js code across multiple files, and to help me understand which parts of the Example 05 structure needed to change versus which could stay the same.

Summary of Interaction: The tool provided a great code for the calm night background and the floating when I described the camera concept I had in mind. Claude read through example file and understood the architecture. I asked follow-up prompts to refine specific parts to meet my ideation extending the discovery flash overlay to stay longer, adding a breathing text prompt, and I adjusting camera speed values. Claude explained what each change did so I understood the reasoning, not just the output.

Human Decision Point(s): I came up with the core idea of a calm breathing exercise screen combined with hidden star-like symbol objects placed across the world. Each symbol, when clicked, would pause the camera and show a short reflective message something to encourage the user to take a break from hectic screen time and follow a calm breathing rhythm alongside the drifting camera.

Integrity & Verification Note: Claude generated all code structure and syntax. I reviewed every file Claude produced, tested them in Live Server, and made manual adjustments to variable values after seeing how it felt in motion.

Scope of GenAI Use: GenAi was NOT used for any creative, brainstorming any ideas (like the calmness, calm night background and breathing gems with hidden messages) GenAI was not used to write this process document,

Limitations or Misfires: The flash overlay faded too quickly in the first version (80 frames). I asked Claude to tell me where to change it and add the breathing text, which required a second prompt and some back and forth on the fade timing. Also, some symbol positions in levels.json were slightly off relative to the platform heights, I manually adjusted the Y values after testing

Summary of Process (Human + Tool)

- My role was deciding the emotional direction, choosing which messages felt right for each symbol, adjusting values like BASE_SPEED and flashTimer to get the pacing to feel genuinely slow and intentional, and pushing back when something felt too fast or too visually busy.

Decision Points & Trade-offs

- Removed player entirely rather than keeping it optional since having a player on screen broke the meditative tone completely
- Used Perlin noise for speed instead of a fixed constant fixed speed felt robotic and emotionally flat

Verification and Judgement

I tested the sketch in Live Server after every significant change. I clicked every symbol to verify the overlay appeared correctly and the message was readable. I adjusted base_speed after seeing the camera felt slightly rushed, again after realising too slowly made the world feel stagnant.

Limitations, Dead Ends, or Open Questions

- There was no such as major limitation, but the background could have been more enhanced maybe with a shooting star showing a ray of hope and calmness effects.

Appendix

Prompt 1: *"Create a reflective or an calm meditative camera experience that scrolls through a world larger than the screen... I want the background to showcase a starry calm night. Use pacing and motion to evoke an calm emotion that priorities breathing exercise. Make sure to add the bonus as well. hide small interactive symbols or objects for the camera to discover. I have added the Week 5 example code that I will be using in VS Code, give me proper code for where each section should be updated."*

Prompt 2: *"Can you please edit the hidden objects as an breathing (inhale-exhale) circle such that when the camera stops and user clicks on it, there is a nice sweet short message pop up and then the camera moves calmly onto the next circle.*

Prompt 3:*"Once the circle object is clicked, can you keep the screen that pops up for a little longer so the users can have more time to read rather than glimpse, also add a text saying or explaining a calm breathing emotion"*

Prompt 4:*"Can you tell me what value to update to change the camera/screen movement speed"*

Prompt 5: "I want to make the background stars more bigger and easier to notice, can you highlight what variable I need to change on code and give me all final code in proper segement files, thank you"

Final Code Produced

sketch.js

```
const VIEW_W = 900;
const VIEW_H = 500;

let allLevelsData;
let level;
let cam;
```

```
let scrollT     = 0;
let scrollSpeed = 0.4;
let targetSpeed = 0.4;
let pauseTimer  = 0;



const BASE_SPEED   = 0.4;
const MAX_SPEED    = 1.1;
const PAUSE_FRAMES = 110;



let discovered = [];
let flashTimer = 0;
let flashMsg   = "";



function preload() {
  allLevelsData = loadJSON("levels.json");
}



function setup() {
  createCanvas(VIEW_W, VIEW_H);
  textFont("Georgia, serif");
  cam = new Camera2D(VIEW_W, VIEW_H);
  cam.x = 0;
  cam.y = 0;
  level = LevelLoader.fromLevelsJson(allLevelsData, 0);
}



function draw() {
  scrollT += 0.005;



  if (pauseTimer > 0) {
    pauseTimer--;
    scrollSpeed = lerp(scrollSpeed, 0, 0.06);
  } else {
    const n = noise(scrollT);
    targetSpeed = map(n, 0, 1, BASE_SPEED * 0.25, MAX_SPEED);
    scrollSpeed = lerp(scrollSpeed, targetSpeed, 0.018);
  }



  cam.scrollAutoX(scrollSpeed);
  cam.clampToWorld(level.w, level.h);
```

```
if (cam.x >= level.w - VIEW_W - 1) {
  cam.x = 0;
  scrollT = 0;
}



const breatheY = sin(frameCount * 0.007) * 20;
cam.y = lerp(cam.y, breatheY, 0.025);



const nearIdx = level.getNearSymbol(cam.x + VIEW_W / 2, 200);
if (nearIdx !== -1 && !discovered.includes(nearIdx) && pauseTimer === 0) {
  pauseTimer = PAUSE_FRAMES;
}



cam.begin();
  level.drawParallaxBg(cam.x);
  level.drawWorld();
  level.drawHiddenSymbols(discovered, frameCount);
cam.end();



if (flashTimer > 0) {
  let a;
  if (flashTimer > 180) {
    a = map(flashTimer, 220, 180, 0, 210);
  } else if (flashTimer < 40) {
    a = map(flashTimer, 40, 0, 210, 0);
  } else {
    a = 210;
  }



  noStroke();
  fill(245, 238, 210, a);
  rect(0, 0, width, height);



  fill(70, 55, 35, a);
  textAlign(CENTER, CENTER);
  textFont("Georgia, serif");
  textSize(22);
  textStyle(ITALIC);
  text(flashMsg, width / 2, height / 2 - 60);
  textStyle(NORMAL);



  stroke(70, 55, 35, a * 0.5);
  strokeWeight(1);
```

```
    line(width / 2 - 120, height / 2 - 30, width / 2 + 120, height / 2 - 30);
    noStroke();



    const breathePulse = sin(frameCount * 0.05);
    const breatheAlpha = map(breathePulse, -1, 1, a * 0.5, a);



    fill(100, 80, 55, breatheAlpha);
    textSize(14);
    text("breathe in  ...  hold  ...  breathe out", width / 2, height / 2 + 10);



    fill(70, 55, 35, a * 0.65);
    textSize(12);
    text("take a slow breath and let this land", width / 2, height / 2 + 40);



    const circleSize = map(breathePulse, -1, 1, 18, 26);
    noFill();
    stroke(100, 80, 55, a * 0.4);
    strokeWeight(1.2);
    ellipse(width / 2, height / 2 + 90, circleSize, circleSize);
    noStroke();



    textAlign(LEFT, BASELINE);
    flashTimer--;
  }



  drawVignette();



  fill(255, 255, 255, 140);
  noStroke();
  textSize(12);
  textFont("Georgia, serif");
  textAlign(LEFT, BASELINE);
  text("click glowing symbols to discover", 16, height - 14);
  text("discovered: " + discovered.length + " / " + level.symbols.length, 16, height -
30);
}



function mousePressed() {
  const wx = mouseX + cam.x;
  const wy = mouseY + cam.y;
  const idx = level.clickSymbol(wx, wy, discovered);
```

```
  if (idx !== -1) {
    discovered.push(idx);
    flashMsg   = level.symbols[idx].message;
    flashTimer = 220;
    pauseTimer = PAUSE_FRAMES * 4;
  }
}



function drawVignette() {
  noStroke();
  const g = drawingContext.createRadialGradient(
    width / 2, height / 2, height * 0.15,
    width / 2, height / 2, height * 0.9
  );
  g.addColorStop(0, "rgba(0,0,0,0)");
  g.addColorStop(1, "rgba(0,0,0,0.5)");
  drawingContext.fillStyle = g;
  drawingContext.fillRect(0, 0, width, height);
}
```

Camera2D.js

```
class Camera2D {
  constructor(viewW, viewH) {
    this.viewW = viewW;
    this.viewH = viewH;
    this.x = 0;
    this.y = 0;
  }



  followSideScrollerX(targetX, lerpAmt) {
    const desired = targetX - this.viewW / 2;
    this.x = lerp(this.x, desired, lerpAmt);
  }



  scrollAutoX(speed) {
    this.x += speed;
  }



  clampToWorld(worldW, worldH) {
    const maxX = max(0, worldW - this.viewW);
    const maxY = max(0, worldH - this.viewH);
    this.x = constrain(this.x, 0, maxX);
    this.y = constrain(this.y, 0, maxY);
  }



  begin() {
    push();
```

```
    translate(-this.x, -this.y);
  }



  end() {
    pop();
  }
}
```

WorldLevel.js

```
class WorldLevel {
  constructor(levelJson) {
    this.name  = levelJson.name ?? "Level";
    this.theme = Object.assign(
      { bg: "#F0F0F0", platform: "#C8C8C8", blob: "#1478FF" },
      levelJson.theme ?? {}
    );
    this.gravity = levelJson.gravity ?? 0.65;
    this.jumpV   = levelJson.jumpV   ?? -11.0;
    this.camLerp = levelJson.camera?.lerp ?? 0.12;
    this.w      = levelJson.world?.w      ?? 2400;
    this.h      = levelJson.world?.h      ?? 500;
    this.deathY = levelJson.world?.deathY ?? this.h + 200;
    this.start  = Object.assign({ x: 80, y: 220, r: 26 }, levelJson.start ?? {});
    this.platforms = (levelJson.platforms ?? []).map(
      (p) => new Platform(p.x, p.y, p.w, p.h)
    );
    this.symbols = (levelJson.symbols ?? []).map((s) => ({
      x: s.x, y: s.y, r: s.r ?? 18,
      glyph: s.glyph ?? "✦",
      message: s.message ?? "...",
      color: s.color ?? "#FFD580",
    }));
  }



  drawWorld() {
    push();
    rectMode(CORNER);
    noStroke();
    fill(this.theme.platform);
    for (const p of this.platforms) rect(p.x, p.y, p.w, p.h);
    pop();
  }



  drawParallaxBg(camX) {
    const W = this.w;
    const H = this.h;
    const topCol   = color(this.theme.skyTop    ?? "#07071a");
    const horizCol = color(this.theme.skyHorizon ?? "#2b1a3a");
    for (let y = 0; y < H; y++) {
      stroke(lerpColor(topCol, horizCol, y / H));
```

```
    line(0, y, W, y);
  }
  noStroke();
  randomSeed(42);
  fill(255, 255, 255, 90);
  for (let i = 0; i < 120; i++) {
    const sx = (random(W) - camX * 0.05) % W;
    const sy = random(H * 0.6);
    ellipse(sx, sy, random(0.5, 2.2));
  }
  randomSeed();
  this._drawHills(camX, 0.30, H * 0.62, H * 0.30,
                  color(this.theme.hillFar  ?? "#2d1b4e"), 18);
  this._drawHills(camX, 0.55, H * 0.72, H * 0.22,
                  color(this.theme.hillMid  ?? "#3a2460"),  9);
  this._drawHills(camX, 0.80, H * 0.82, H * 0.14,
                  color(this.theme.hillNear ?? "#1e3a2f"),  5);
}



_drawHills(camX, parallax, baseY, amplitude, col, seed) {
  noStroke();
  fill(col);
  const offset = camX * parallax;
  beginShape();
  vertex(0, this.h + 10);
  for (let x = 0; x <= this.w; x += 6) {
    const n = noise(seed * 100 + (x + offset) * 0.0018);
    vertex(x, baseY - n * amplitude);
  }
  vertex(this.w, this.h + 10);
  endShape(CLOSE);
}



drawHiddenSymbols(discovered, frameCount) {
  for (let i = 0; i < this.symbols.length; i++) {
    const s     = this.symbols[i];
    const found = discovered.includes(i);
    const col   = color(s.color);
    if (found) {
      noFill();
      stroke(col);
      strokeWeight(1.5);
      ellipse(s.x, s.y, s.r * 2 + (frameCount * 0.4) % 60,
                        s.r * 2 + (frameCount * 0.4) % 60);
      noStroke();
      fill(red(col), green(col), blue(col), 60);
      textAlign(CENTER, CENTER);
      textSize(s.r * 1.2);
      text(s.glyph, s.x, s.y);
    } else {
      const pulse  = sin(frameCount * 0.06 + i * 1.3);
      const glow   = map(pulse, -1, 1, 0.5, 1.0);
      const radius = s.r + pulse * 4;
      noStroke();
      fill(red(col), green(col), blue(col), 25 * glow);
      ellipse(s.x, s.y, radius * 3.5, radius * 3.5);
      fill(red(col), green(col), blue(col), 55 * glow);
      ellipse(s.x, s.y, radius * 2, radius * 2);
```

```
        fill(red(col), green(col), blue(col), 160 * glow);
        ellipse(s.x, s.y, radius, radius);
        fill(255, 255, 255, 200 * glow);
        textAlign(CENTER, CENTER);
        textSize(s.r * 0.95);
        text(s.glyph, s.x, s.y);
      }
    }
    textAlign(LEFT, BASELINE);
    strokeWeight(1);
    noStroke();
  }



  getNearSymbol(worldX, range) {
    for (let i = 0; i < this.symbols.length; i++) {
      if (abs(this.symbols[i].x - worldX) < range) return i;
    }
    return -1;
  }



  clickSymbol(wx, wy, alreadyFound) {
    for (let i = 0; i < this.symbols.length; i++) {
      if (alreadyFound.includes(i)) continue;
      const s = this.symbols[i];
      if (dist(wx, wy, s.x, s.y) < s.r + 14) return i;
    }
    return -1;
  }
}
```

Levels.json

```
{
  "schemaVersion": 1,
  "levels": [
    {
      "name": "Dreamscape",
      "theme": {
        "bg": "#0d0d1a", "platform": "#2a3d5a",
        "skyTop": "#07071a", "skyHorizon": "#2b1a3a",
        "hillFar": "#1a1035", "hillMid": "#22183d", "hillNear": "#0e2820"
      },
      "gravity": 0.65, "jumpV": -11.0,
      "camera": { "lerp": 0.06 },
      "world": { "w": 5000, "h": 500, "deathY": 900 },
      "start": { "x": 80, "y": 300, "r": 26 },
      "platforms": [
        { "x": 0,    "y": 460, "w": 5000, "h": 40 },
        { "x": 200,  "y": 370, "w": 160,  "h": 12 },
        { "x": 480,  "y": 320, "w": 120,  "h": 12 },
        { "x": 680,  "y": 270, "w": 100,  "h": 12 },
        { "x": 900,  "y": 350, "w": 200,  "h": 12 },
        { "x": 1150, "y": 300, "w": 130,  "h": 12 },
        { "x": 1380, "y": 240, "w": 80,   "h": 12 },
```

```
        { "x": 1550, "y": 310, "w": 220,  "h": 12 },
        { "x": 1820, "y": 270, "w": 150,  "h": 12 },
        { "x": 2100, "y": 350, "w": 300,  "h": 12 },
        { "x": 2460, "y": 290, "w": 120,  "h": 12 },
        { "x": 2680, "y": 240, "w": 90,   "h": 12 },
        { "x": 2900, "y": 330, "w": 180,  "h": 12 },
        { "x": 3150, "y": 280, "w": 140,  "h": 12 },
        { "x": 3400, "y": 360, "w": 260,  "h": 12 },
        { "x": 3720, "y": 310, "w": 110,  "h": 12 },
        { "x": 3950, "y": 260, "w": 100,  "h": 12 },
        { "x": 4200, "y": 340, "w": 200,  "h": 12 },
        { "x": 4500, "y": 390, "w": 300,  "h": 12 }
      ],
      "symbols": [
        { "x": 420,  "y": 295, "r": 20, "glyph": "◯", "color": "#a8d8ea",
          "message": "stillness is not emptiness" },
        { "x": 870,  "y": 220, "r": 18, "glyph": "✦", "color": "#f9d4a0",
          "message": "breathe in what you cannot name" },
        { "x": 1340, "y": 205, "r": 22, "glyph": "△", "color": "#c3aed6",
          "message": "the peak is just another place to rest" },
        { "x": 1780, "y": 240, "r": 19, "glyph": "✧", "color": "#80ced7",
          "message": "distance is only felt when you stop moving" },
        { "x": 2420, "y": 260, "r": 21, "glyph": "☽", "color": "#ffe0b2",
          "message": "even the moon forgets its shape sometimes" },
        { "x": 2850, "y": 300, "r": 17, "glyph": "∞", "color": "#b2f7ef",
          "message": "loops are not failure — they are return" },
        { "x": 3380, "y": 250, "r": 20, "glyph": "✧", "color": "#ffd6e0",
          "message": "you are allowed to be exactly here" },
        { "x": 3900, "y": 225, "r": 18, "glyph": "◈", "color": "#d4f1c0",
          "message": "notice the quiet between the notes" },
        { "x": 4450, "y": 360, "r": 22, "glyph": "❋", "color": "#f0c4d4",
          "message": "every journey ends where it always began" }
      ]
    }
  ]
}
```

Index.html

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>Meditative Camera — Week 5</title>
<link rel="stylesheet" href="style.css" />
<script src="libraries/p5.min.js"></script>
<script src="Platform.js"></script>
<script src="WorldLevel.js"></script>
<script src="Camera2D.js"></script>
<script src="LevelLoader.js"></script>
<script src="sketch.js"></script>
</head>
<body><main></main></body>
</html>
```