

Process & Decision Documentation

Project/Assignment Overview + Decisions

This document covers the design and development process for Passing Fields; a reflective camera scroll experience built in p5.js. My goal was to use pacing and motion to evoke emotion rather than traditional game mechanics.

The key decision I made was to give players control over how they experience the scroll by defaulting to manual arrow key navigation while offering an autoscroll toggle. This lets the experience work as both a passive meditative piece and an active exploration.

Role-Based Process Evidence

- Started with the base camera scroll
- Built out the layered world with sky, hills, trees, and floating petals
- Added the four bonus symbols (sun, leaf, star, moon) with pulse/glow
- Iterated on the day-to-night transition
- Added a start screen explaining controls and an autoscroll toggle button

Entry Header

Name: Jason Pham

Role(s): Lead

Primary responsibility for this work: Jason Pham

Goal of Work Session

My goal of this work session was to build a meditative and immersive scrolling camera experience.

Tools, Resources, or Inputs Used

- p5.js
- Web browser for testing and debugging
- Lecture material on game mechanics and affordances
- Informal self-playtesting
- GenAI (coding support)

GenAI Documentation

If GenAI was used (keep each response as brief as possible):

Date Used: During development of this Side Quest

Tool Disclosure: Claude (Opus 4.5)

Purpose of Use:

Used GenAI to assist with:

- initial set up the camera scroll and parallax layers
- build the day-night transition with easing
- add symbol discovery and pulse/glow effects
- create the start screen
- handle switching between manual and autoscroll modes

Summary of Interaction:

GenAI helped:

- scaffold movement
- collision logic
- conditional behavior

Human Decision Point(s):

I went and modified the generated code to try and better fit the intended pacing and mechanics. I manually adjusted speed values, enemy behavior, and fixed some of the bugs when GenAI solutions didn't work as intended.

Integrity & Verification Note:

All generated code was tested in the browser and verified through repeated playtesting.

Scope of GenAI Use:

GenAI did not design the game concept, emotional framing, the symbol ordering logic, or the transition.

Limitations or Misfires:

GenAI could not account for emotional pacing or difficulty balance, requiring manual tuning.

Summary of Process (Human + Tool):

During the process of implementing mechanics, I would test them all through gameplay, trying to identify issues, and making sure to refine values. While GenAI supported early coding efficiency, I guided the emotional intent, balance, and iterations.

Decision Points & Trade-offs

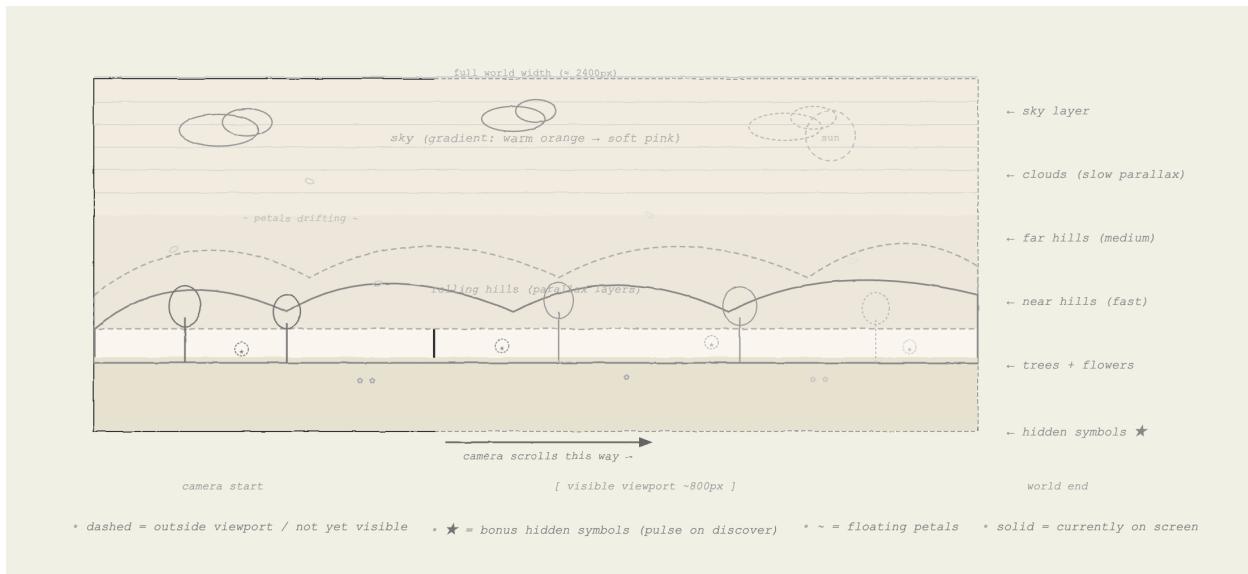
- Choose to default to manual controls over autoscroll to give players agency over their own pace
- Tied the day-night transition to camera position rather than a timer so the shift feels discovered
- Kept the autoscroll toggle always visible so players can switch modes without restarting

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

Pre design of what I wanted to do:



Create and simulates a slow, meditative camera scrolling horizontally through a nature world that is 3x wider than the canvas (canvas size 800x400).

The camera should drift automatically from left to right using a camX variable and translate().

The world should have a layered parallax feel with a gradient sky background, rolling hills or clouds in the midground, and small details like trees, flowers, or grass in the foreground. Use soft, warm pastel colors to create a calm emotional tone. Add floating particles (petals or fireflies) that drift slowly across the screen.

For the bonus, hide 4 small glowing symbols scattered through the world. When the camera scrolls to reveal them on screen, they should pulse or glow to signal discovery.

Keep everything in a single sketch.js file using only p5.js. No external libraries. Include comments explaining each section.

Show more

[Read sketch.js](#)

[Read index.html](#)

Good, p5.js is loaded locally. Now I'll write the full sketch.

[Write sketch.js](#)

428 lines

```
// =====  
// MEDITATIVE NATURE SCROLL  
// Canvas: 800 × 400 | World: 2400 wide (3× canvas)  
// p5.js only — no external libraries  
// =====  
  
// — WORLD & CAMERA ——————  
const WORLD_W = 2400; // total world width (3× canvas)  
const CANVAS_W = 800;  
const CANVAS_H = 400;  
const SPEED = 0.5; // camera drift speed in px/frame
```

```
let camX = 0;      // world-space left edge of the camera

// —— TERRAIN PROFILES ——————
// Pre-built once in setup() using overlapping sine waves.
// Each is an array of {x, y} world-space points.
let farPts = []; // distant lavender hills (parallax 0.35)
let midPts = []; // sage-green mid hills (parallax 0.62)
let gndPts = []; // near meadow ground (parallax 1.0)

// —— FLOATING PETALS ——————
let petals = [];

// —— HIDDEN SYMBOLS (BONUS) ——————
// 4 glowing symbols scattered through the world.
// Each pulses when the camera reveals them.
let symbols = [];

// =====
// SETUP
// =====
function setup() {
  createCanvas(CANVAS_W, CANVAS_H);

  // Build terrain curves — overlapping sine waves give organic hills
  for (let x = 0; x <= WORLD_W; x += 6) {
```

```

farPts.push(createVector(x,
 212 + sin(x * 0.003)    * 52
 + sin(x * 0.007 + 1.2) * 26
));
midPts.push(createVector(x,
 270 + sin(x * 0.0045 + 2) * 34
 + sin(x * 0.009 + 4.5) * 15
));
gndPts.push(createVector(x,
 316 + sin(x * 0.006 + 1) * 9
));
}

// Create 55 drifting petal particles scattered across the world
for (let i = 0; i < 55; i++) {
  petals.push({
    wx: random(WORLD_W),    // world-space x
    y: random(80, 345),    // screen y
    sz: random(3, 7.5),    // petal size
    vx: random(0.18, 0.65), // rightward drift speed
    ph: random(TWO_PI),    // phase offset for vertical float
    ang: random(TWO_PI),   // rotation angle
    aSpd: random(-0.025, 0.025), // rotation speed
    r: random(230, 255),   // soft pink/peach tones
    g: random(148, 218),
    b: random(182, 234),
  });
}

```

```

    a: random(140, 210)

});

}

// Place 4 hidden glowing symbols at specific world positions
symbols = [
  { wx: 480, wy: 192, type: 'star', ph: 0      },
  { wx: 1110, wy: 180, type: 'moon', ph: PI      },
  { wx: 1670, wy: 202, type: 'leaf', ph: HALF_PI  },
  { wx: 2210, wy: 188, type: 'sun', ph: PI + HALF_PI }
];

}

// =====
// DRAW — called every frame
// =====

function draw() {
  // Advance camera and loop seamlessly at the right edge
  camX = (camX + SPEED) % (WORLD_W - CANVAS_W);

  // — Layer 0: Sky gradient — fixed, no parallax ————
  drawSky();

  // — Layer 1: Distant clouds — very slow parallax 0.15 —
  push();
  translate(-camX * 0.15, 0);
}

```

```
drawClouds();

pop();

// — Layer 2: Far hills — parallax 0.35 ——————
push();
translate(-camX * 0.35, 0);
drawFarHills();
pop();

// — Layer 3: Mid hills — parallax 0.62 ——————
push();
translate(-camX * 0.62, 0);
drawMidHills();
pop();

// — Layer 4: Foreground — full camera speed (1.0) ——————
// Near ground, trees, flowers, petals, and symbols all live here.
push();
translate(-camX, 0);
drawNearGround();
drawTrees();
drawFlowers();
drawPetals();
drawSymbols();
pop();
}
```

```
// =====  
  
// SKY — vertical warm-pastel gradient  
  
// Top: soft lavender-blue → Bottom: warm peachy-cream  
  
// =====  
  
function drawSky() {  
  
    noStroke();  
  
    for (let y = 0; y < CANVAS_H; y++) {  
  
        let t = y / CANVAS_H;  
  
        stroke(  
            lerp(185, 255, t), // R: lavender → peach  
            lerp(172, 210, t), // G: cool → warm  
            lerp(230, 178, t) // B: blue → amber  
        );  
  
        line(0, y, CANVAS_W, y);  
    }  
  
    noStroke();  
}  
  
// =====  
  
// CLOUDS — fluffy ellipse clusters with a rosy blush  
  
// =====  
  
function drawClouds() {  
  
    // Cloud positions are spaced across the full world width.  
  
    // Because parallax is 0.15, the effective viewing window  
  
    // is much narrower, so clouds appear to fill the sky nicely.
```

```
const clouds = [
  {x: 150, y: 65, s: 1.2 },
  {x: 490, y: 52, s: 0.9 },
  {x: 830, y: 70, s: 1.4 },
  {x: 1170, y: 56, s: 1.0 },
  {x: 1500, y: 68, s: 1.15 },
  {x: 1840, y: 50, s: 0.85 },
  {x: 2180, y: 73, s: 1.1 },
];

noStroke();
for (let c of clouds) {
  // Main cloud body — three overlapping ellipses
  fill(255, 245, 250, 195);
  ellipse(c.x,      c.y,      80*c.s, 38*c.s);
  ellipse(c.x - 34*c.s, c.y + 10*c.s, 56*c.s, 30*c.s);
  ellipse(c.x + 38*c.s, c.y + 8*c.s, 60*c.s, 28*c.s);
  // Rosy blush highlight on top
  fill(255, 220, 235, 55);
  ellipse(c.x, c.y - 5, 60*c.s, 22*c.s);
}
}

// =====
// FAR HILLS — muted lavender silhouette
// =====
```

```
function drawFarHills() {  
    // Filled hill shape from precomputed profile down to bottom of canvas  
    fill(205, 188, 225);  
    noStroke();  
    beginShape();  
    vertex(0, CANVAS_H);  
    for (let p of farPts) vertex(p.x, p.y);  
    vertex(WORLD_W, CANVAS_H);  
    endShape(CLOSE);  
  
    // Subtle rim-light along the ridge  
    stroke(225, 212, 240, 110);  
    strokeWeight(1.5);  
    noFill();  
    beginShape();  
    for (let p of farPts) vertex(p.x, p.y - 2);  
    endShape();  
    noStroke();  
}  
  
// ======  
// MID HILLS — sage-green rolling meadow  
// ======  
function drawMidHills() {  
    fill(162, 204, 170);  
    noStroke();
```

```
beginShape();
vertex(0, CANVAS_H);
for (let p of midPts) vertex(p.x, p.y);
vertex(WORLD_W, CANVAS_H);
endShape(CLOSE);

}

// =====
// NEAR GROUND — rich meadow base strip (foreground layer)
// =====

function drawNearGround() {
fill(130, 182, 142);
noStroke();
beginShape();
vertex(0, CANVAS_H);
for (let p of gndPts) vertex(p.x, p.y);
vertex(WORLD_W, CANVAS_H);
endShape(CLOSE);

}

// =====
// TREES — layered foliage blobs, snapped to ground surface
// =====

function drawTrees() {
const treeXs = [
85, 255, 435, 615, 800,
```

```
1015, 1195, 1385, 1570, 1755,  
1945, 2135, 2325  
];  
  
for (let tx of treeXs) {  
  
    let gY = groundY(tx);  
  
    // Vary height and width using Perlin noise for organic feel  
  
    let h = 58 + noise(tx * 0.01) * 38;  
    let w = 46 + noise(tx * 0.02 + 5) * 18;  
  
    // Trunk  
  
    fill(148, 108, 78);  
    noStroke();  
    rect(tx - 5, gY - h + 12, 10, h);  
  
    // Three overlapping foliage ellipses create a soft, layered crown  
  
    fill(108, 162, 122, 218);  
    ellipse(tx, gY - h - 4, w, w * 0.95);  
    fill(92, 150, 110, 200);  
    ellipse(tx - 14, gY - h + 9, w * 0.8, w * 0.8);  
    fill(122, 175, 136, 200);  
    ellipse(tx + 11, gY - h + 11, w * 0.74, w * 0.74);  
}  
}  
  
// =====
```

```

// FLOWERS + GRASS BLADES — foreground botanical detail
// =====

function drawFlowers() {
    // Grass blades: two angled lines per clump, spaced across world
    strokeWeight(1.2);

    for (let x = 0; x < WORLD_W; x += 18) {
        let gy = groundY(x);

        let bh = 7 + noise(x * 0.14) * 10;
        stroke(90, 152, 100, 165);

        line(x, gy, x - 2, gy - bh);
        line(x + 7, gy, x + 9, gy - bh * 0.75);
    }

    noStroke();
}

// Flowers: one per region, each with a distinct pastel hue
const flowers = [
    { x: 140, c: [255, 172, 185] }, // blush pink
    { x: 305, c: [255, 228, 142] }, // butter yellow
    { x: 485, c: [212, 172, 255] }, // soft violet
    { x: 665, c: [255, 188, 200] }, // rose
    { x: 865, c: [255, 235, 152] }, // warm yellow
    { x: 1055, c: [188, 172, 255] }, // periwinkle
    { x: 1235, c: [255, 198, 168] }, // apricot
    { x: 1425, c: [172, 218, 255] }, // sky blue
    { x: 1615, c: [255, 172, 212] }, // carnation
    { x: 1805, c: [255, 225, 152] }, // golden
]

```

```
{ x: 1995, c: [192, 255, 192] }, // mint
{ x: 2195, c: [255, 192, 225] }, // cotton candy
{ x: 2365, c: [255, 240, 158] }, // lemon
};

for (let f of flowers) {
  let gy = groundY(f.x);

  // Stem
  stroke(92, 145, 80);
  strokeWeight(1.4);
  line(f.x, gy, f.x, gy - 20);
  noStroke();

  // 6 petals arranged in a ring
  fill(f.c[0], f.c[1], f.c[2], 200);
  for (let i = 0; i < 6; i++) {
    let a = (i / 6) * TWO_PI;
    ellipse(f.x + cos(a) * 6, (gy - 20) + sin(a) * 6, 8, 8);
  }

  // Bright center
  fill(255, 242, 100);
  ellipse(f.x, gy - 20, 7, 7);
}
}
```

```
// =====  
  
// PETALS — drifting floating particles in world space  
// =====  
  
function drawPetals() {  
  
    noStroke();  
  
    for (let p of petals) {  
  
        // Update: drift right, float vertically on a sine wave, spin slowly  
  
        p.wx += p.vx;  
  
        p.y += sin(frameCount * 0.018 + p.ph) * 0.35;  
  
        p.ang += p.aSpd;  
  
        // Wrap around the world  
  
        if (p.wx > WORLD_W) p.wx = 0;  
  
        if (p.y > CANVAS_H - 25) p.y = 85;  
  
        // Cull: skip petals that are off-screen (world-space check)  
  
        if (p.wx < camX - 15 || p.wx > camX + CANVAS_W + 15) continue;  
  
        // Draw an elongated oval rotated to look like a drifting petal  
  
        push();  
  
        translate(p.wx, p.y);  
  
        rotate(p.ang);  
  
        fill(p.r, p.g, p.b, p.a);  
  
        ellipse(0, 0, p.sz * 2.3, p.sz);  
  
        pop();  
    }  
}
```

```
}

}

// =====

// SYMBOLS — 4 hidden glowing discoveries (BONUS)

// Each symbol pulses when the camera reveals it on screen.

// =====

function drawSymbols() {

  for (let s of symbols) {

    // Advance the pulse phase continuously

    s.ph += 0.055;

    // Only draw when within screen bounds (world-space check)

    let sx = s.wx - camX;

    if (sx < -60 || sx > CANVAS_W + 60) continue;

    // pulse: 0 → 1 smooth oscillation

    let pulse = (sin(s.ph) + 1) * 0.5;

    // Radial glow — concentric rings fade outward

    noStroke();

    let maxR = 20 + pulse * 14;

    for (let r = maxR; r > 0; r -= 2.5) {

      let a = map(r, maxR, 0, 0, 65 + pulse * 115);

      fill(255, 228, 115, a);

      ellipse(s.wx, s.wy, r * 2, r * 2);

    }

  }

}
```

```
}
```

```
// Symbol fill and outline, alpha-boosted by pulse

let sa = 175 + pulse * 80;

fill(255, 245, 158, sa);

stroke(255, 205, 55, sa);

strokeWeight(1.5);

if (s.type === 'star') drawStarSym(s.wx, s.wy, 8, 18, 5);

else if (s.type === 'moon') drawMoonSym(s.wx, s.wy, 13, pulse);

else if (s.type === 'leaf') drawLeafSym(s.wx, s.wy, 13, pulse);

else if (s.type === 'sun') drawSunSym (s.wx, s.wy, 10, pulse);

}

}
```

```
// — Symbol: 5-point star
```

```
function drawStarSym(x, y, r1, r2, n) {

beginShape();

for (let i = 0; i < n * 2; i++) {

let r = (i % 2 === 0) ? r2 : r1;

let a = (i / (n * 2)) * TWO_PI - HALF_PI;

vertex(x + cos(a) * r, y + sin(a) * r);

}

endShape(CLOSE);

}
```

```
// — Symbol: crescent moon ——————  
// Carved by overdrawing a smaller ellipse in a contrasting hue.  
  
function drawMoonSym(x, y, r, pulse) {  
  
    noStroke();  
  
    fill(255, 242, 185, 200 + pulse * 55);  
  
    ellipse(x, y, r * 2, r * 2);  
  
    // "Cut out" the crescent shape with a mid-purple disc  
  
    fill(190, 162, 222, 235);  
  
    ellipse(x + r * 0.48, y, r * 1.52, r * 1.72);  
  
}
```

```
// — Symbol: botanical leaf ——————  
  
function drawLeafSym(x, y, r, pulse) {  
  
    fill(158, 225, 158, 200 + pulse * 55);  
  
    stroke(95, 182, 108, 220);  
  
    strokeWeight(1.2);  
  
    // Symmetrical leaf via mirrored bezier curves  
  
    beginShape();  
  
    vertex(x, y - r);  
  
    bezierVertex(x + r, y - r * 0.3, x + r, y + r * 0.3, x, y + r);  
  
    bezierVertex(x - r, y + r * 0.3, x - r, y - r * 0.3, x, y - r);  
  
    endShape(CLOSE);  
  
    // Central midrib vein  
  
    stroke(95, 182, 108, 175);  
  
    line(x, y - r, x, y + r);  
  
}
```

```

// — Symbol: radiant sun ——————  

function drawSunSym(x, y, r, pulse) {  

    noStroke();  

    fill(255, 235, 98, 200 + pulse * 55);  

    ellipse(x, y, r * 2, r * 2);  

    // 8 rays that lengthen slightly with the pulse  

    stroke(255, 215, 65, 175 + pulse * 80);  

    strokeWeight(2);  

    for (let i = 0; i < 8; i++) {  

        let a = (i / 8) * TWO_PI;  

        let i1 = r + 3;  

        let o1 = r + 8 + pulse * 5;  

        line(  

            x + cos(a) * i1, y + sin(a) * i1,  

            x + cos(a) * o1, y + sin(a) * o1  

        );  

    }  

}  

// ======  

// HELPER — sample near-ground Y at any world X  

// ======  

function groundY(wx) {  

    return 316 + sin(wx * 0.006 + 1) * 9;  

}

```

Here's what the sketch does, layer by layer:

Camera system

camX increments by 0.5 px/frame and loops seamlessly with % (WORLD_W - CANVAS_W)

Each layer uses push(); translate(-camX * factor, 0); pop() to create independent parallax speeds

5 parallax layers

Layer	Content	Parallax
0	Sky gradient	0 (fixed)
1	Fluffy clouds	0.15 (slowest)
2	Lavender far hills	0.35
3	Sage-green mid hills	0.62
4	Ground, trees, flowers, petals, symbols	1.0 (full speed)

Visual details

Sky: per-line gradient loop from lavender-blue at top to warm peach at bottom

Hills: precomputed from two overlapping sine waves per layer for organic, non-repeating shapes

Trees: 13 trees with Perlin-noise height/width variation, three foliage ellipses each for depth

Flowers: 13 varieties in distinct pastel hues with 6 petals each; grass blades fill the gaps

Petals: 55 particles drifting right with slow vertical sine oscillation and lazy rotation; culled off-screen for performance

Bonus — 4 glowing symbols

Star at x=480, Moon at x=1110, Leaf at x=1670, Sun at x=2210

Each has a ph (phase) that pulses continuously; when s.wx - camX lands on-screen, concentric glow rings and alpha all surge with the pulse

make it so the bonus symbols showcase in order of sun, leaf, star, moon.

let the leaf be on the ground, make the sun on the left side of the screen, make the moon on the right side.

add a transition from this afternoon to night, should transition faster closer around the leaf, smoothen the transition so it eases.

Read sketch.js

Now I have a full picture of the existing code. Let me rewrite it with all the changes — symbol reorder, leaf on ground, sun left / moon right, and the eased day-to-night sky transition.

Write sketch.js

593 lines

```
// =====  
// MEDITATIVE NATURE SCROLL — AFTERNOON TO NIGHT  
// Canvas: 800 × 400 | World: 2400 wide (3× canvas)  
// Narrative: camera drifts left→right through the day.  
// Symbols in discovery order: Sun → Leaf → Star → Moon  
// Sun (wx=150) — left side of screen, full afternoon  
// Leaf (wx=950) — resting on the ground, golden hour  
// Star (wx=1500) — dusk sky, stars emerging  
// Moon (wx=2300) — right side of screen, deep night  
// p5.js only — no external libraries  
// =====
```

```
// —— WORLD & CAMERA ——————  
const WORLD_W = 2400;  
const CANVAS_W = 800;  
const CANVAS_H = 400;  
const SPEED = 0.5; // camera drift (px/frame)  
  
let camX = 0; // world-space left edge of camera  
  
// —— TIME OF DAY ——————  
// tod: 0 = bright afternoon, 1 = deep night  
// Computed each frame from camX with a non-linear easing  
// so the transition accelerates sharply around the Leaf symbol.  
let tod = 0;  
  
// —— TERRAIN PROFILES ——————  
let farPts = []; // distant lavender hills (parallax 0.35)  
let midPts = []; // sage-green mid hills (parallax 0.62)  
let gndPts = []; // near meadow ground (parallax 1.0)  
  
// —— FLOATING PETALS / FIREFLIES ——————  
let petals = [];  
  
// —— STARS (screen-space, fade in at dusk) ——————  
let stars = [];
```

```
// — HIDDEN SYMBOLS ——————  
  
let symbols = [];  
  
// ======  
  
// SETUP  
// ======  
  
function setup() {  
  
    createCanvas(CANVAS_W, CANVAS_H);  
  
    // Build terrain from overlapping sine waves — organic, non-repeating  
    for (let x = 0; x <= WORLD_W; x += 6) {  
  
        farPts.push(createVector(x,  
            212 + sin(x * 0.003) * 52  
            + sin(x * 0.007 + 1.2) * 26  
        ));  
  
        midPts.push(createVector(x,  
            270 + sin(x * 0.0045 + 2) * 34  
            + sin(x * 0.009 + 4.5) * 15  
        ));  
  
        gndPts.push(createVector(x,  
            316 + sin(x * 0.006 + 1) * 9  
        ));  
    }  
  
    // Floating petal particles distributed across the world  
    for (let i = 0; i < 55; i++) {
```

```
petals.push({  
    wx: random(WORLD_W),  
    y: random(80, 345),  
    sz: random(3, 7.5),  
    vx: random(0.18, 0.65),  
    ph: random(TWO_PI),  
    ang: random(TWO_PI),  
    aSpd: random(-0.025, 0.025),  
    r: random(230, 255), // daytime: soft pink-peach  
    g: random(148, 218),  
    b: random(182, 234),  
    a: random(140, 210)  
});  
}  
  
});
```

```
// Stars in screen-space (they don't scroll — sky is infinite)  
for (let i = 0; i < 90; i++) {  
    stars.push({  
        x: random(CANVAS_W),  
        y: random(CANVAS_H * 0.62), // upper sky only  
        sz: random(0.8, 2.4),  
        ph: random(TWO_PI) // twinkle phase offset  
    });  
}  
  
// — 4 hidden glowing symbols ——————
```

```
// Leaf sits on the ground — snap its wy to groundY(950) - radius  
let leafGroundY = 316 + sin(950 * 0.006 + 1) * 9; // ≈ 320
```

```
symbols = [  
    // Sun: first discovery, left side of screen at scroll start  
    { wx: 150, wy: 158, type: 'sun', ph: 0 },  
    // Leaf: resting on the ground, marks the golden-hour pivot  
    { wx: 950, wy: leafGroundY - 14, type: 'leaf', ph: HALF_PI },  
    // Star: dusk area, sky filling with stars  
    { wx: 1500, wy: 188, type: 'star', ph: PI },  
    // Moon: final discovery, right side of screen at scroll end  
    { wx: 2300, wy: 162, type: 'moon', ph: PI+HALF_PI }  
];  
}
```

```
// ======  
// HELPERS  
// ======
```

```
// Classic smooth-step: ease in AND out over [0,1]  
function smoothStep(t){  
    t = constrain(t, 0, 1);  
    return t * t * (3 - 2 * t);  
}
```

```
// Compute time-of-day from camX.
```

```

// Three segments give a slow → FAST → slow easing curve:

// Segment A camX 0→352 (p 0→0.22) tod 0→0.12 very slow, still afternoon
// Segment B camX 352→832 (p 0.22→0.52) tod 0.12→0.88 FAST burst at the Leaf
// Segment C camX 832→1600 (p 0.52→1) tod 0.88→1.0 slow tail into night

function computeTOD() {

    let p = camX / (WORLD_W - CANVAS_W); // 0 → 1

    if (p < 0.22) {

        return lerp(0, 0.12, smoothStep(p / 0.22));

    } else if (p < 0.52) {

        return lerp(0.12, 0.88, smoothStep((p - 0.22) / 0.30));

    } else {

        return lerp(0.88, 1.0, smoothStep((p - 0.52) / 0.48));

    }

}

// Interpolate through an array of [r,g,b] color stops

function lerpStops(stops, t) {

    let n = stops.length - 1;

    let i = min(floor(t * n), n - 1);

    let lt = (t * n) - i;

    return [

        lerp(stops[i][0], stops[i+1][0], lt),

        lerp(stops[i][1], stops[i+1][1], lt),

        lerp(stops[i][2], stops[i+1][2], lt)

    ];

}

```

```
}

// Ground surface Y at any world X (must match gndPts formula)

function groundY(wx) {

    return 316 + sin(wx * 0.006 + 1) * 9;

}
```

```
// Scale an RGB color toward black for night silhouette.

// scale: 1 = full daytime colour, ~0.06 = near-black night

function dn(r, g, b, scale) {

    return [r * scale, g * scale, b * scale];

}
```

```
// =====

// DRAW

// =====

function draw() {

    camX = (camX + SPEED) % (WORLD_W - CANVAS_W);

    tod = computeTOD();

    // — Layer 0: Sky (no parallax — fixed backdrop) ————
    drawSky();

    // — Stars fade in at dusk, screen-space before hills ————
    drawStars();
```

```
// — Layer 1: Clouds (parallax 0.15 — barely drift) ——————
```

```
push();  
translate(-camX * 0.15, 0);  
drawClouds();  
pop();
```

```
// — Layer 2: Far hills (parallax 0.35) ——————
```

```
push();  
translate(-camX * 0.35, 0);  
drawFarHills();  
pop();
```

```
// — Layer 3: Mid hills (parallax 0.62) ——————
```

```
push();  
translate(-camX * 0.62, 0);  
drawMidHills();  
pop();
```

```
// — Layer 4: Full-speed foreground (parallax 1.0) ——————
```

```
push();  
translate(-camX, 0);  
drawNearGround();  
drawTrees();  
drawFlowers();  
drawPetals(); // petals by day, fireflies by night  
drawSymbols();
```

```
pop();  
}  
  
// ======  
// SKY — multi-stop gradient: afternoon → golden hour → dusk → night  
// ======  
  
function drawSky() {  
    noStroke();  
  
    // Sky-top colours at each time-of-day stop  
    const topStops = [  
        [185, 172, 230], // afternoon : soft lavender-blue  
        [255, 152, 75], // golden hr : warm orange  
        [72, 42, 138], // dusk   : deep violet  
        [8, 12, 55]  // night  : navy  
    ];  
  
    // Horizon/bottom colours  
    const botStops = [  
        [255, 210, 178], // afternoon : peachy cream  
        [255, 90, 30], // golden hr : red-orange  
        [178, 68, 115], // dusk   : magenta-rose  
        [18, 13, 62] // night  : dark indigo  
    ];  
  
    let topC = lerpStops(topStops, tod);  
    let botC = lerpStops(botStops, tod);
```

```

for (let y = 0; y < CANVAS_H; y++) {
    let t = y / CANVAS_H;
    stroke(
        lerp(topC[0], botC[0], t),
        lerp(topC[1], botC[1], t),
        lerp(topC[2], botC[2], t)
    );
    line(0, y, CANVAS_W, y);
}

noStroke();

// Sunset horizon glow — warm band near hill-line during transition.

// Uses a sine arch so it appears and disappears smoothly.

if (tod > 0.14 && tod < 0.74) {
    let intensity = sin(map(tod, 0.14, 0.74, 0, PI)); // 0→peak→0
    let horizY = CANVAS_H * 0.52;
    for (let y = CANVAS_H * 0.36; y < CANVAS_H * 0.68; y++) {
        let dy = abs(y - horizY) / (CANVAS_H * 0.16);
        let a = max(0, 1 - dy) * intensity * 95;
        stroke(255, 145, 38, a);
        line(0, y, CANVAS_W, y);
    }
    noStroke();
}

```

```

// =====
// STARS — screen-space; twinkle gently; fade in at dusk
// =====

function drawStars() {
    if (tod < 0.32) return;

    let alpha = map(tod, 0.32, 0.72, 0, 255);
    noStroke();
    for (let s of stars) {
        let tw = (sin(frameCount * 0.038 + s.ph) + 1) * 0.5; // 0→1 twinkle
        fill(245, 248, 255, alpha * (0.62 + tw * 0.38));
        let sz = s.sz + tw * 0.55;
        ellipse(s.x, s.y, sz, sz);
    }
}

// =====
// CLOUDS — fade and tint toward dark night wisps
// =====

function drawClouds() {
    // Opacity drops at night; colour shifts from white-pink to dark blue-grey
    let baseA = lerp(192, 28, tod);
    let cr  = lerp(255, 48, tod);
    let cg  = lerp(245, 42, tod);
    let cb  = lerp(250, 88, tod);
}

```

```

const clouds = [
  {x: 150, y: 65, s: 1.2 },
  {x: 490, y: 52, s: 0.9 },
  {x: 830, y: 70, s: 1.4 },
  {x: 1170, y: 56, s: 1.0 },
  {x: 1500, y: 68, s: 1.15 },
  {x: 1840, y: 50, s: 0.85 },
  {x: 2180, y: 73, s: 1.1 },
];

noStroke();

for (let c of clouds) {
  fill(cr, cg, cb, baseA);
  ellipse(c.x,      c.y,      80*c.s, 38*c.s);
  ellipse(c.x - 34*c.s, c.y + 10*c.s, 56*c.s, 30*c.s);
  ellipse(c.x + 38*c.s, c.y + 8*c.s, 60*c.s, 28*c.s);
  // Rosy blush — only visible in daylight
  let blushA = max(0, lerp(52, -10, tod));
  fill(255, 220, 235, blushA);
  ellipse(c.x, c.y - 5, 60*c.s, 22*c.s);
}

// =====
// FAR HILLS — lavender → black silhouette
// =====

```

```
function drawFarHills() {  
  let ns = lerp(1, 0.07, tod);  
  let [r, g, b] = dn(205, 188, 225, ns);  
  fill(r, g, b);  
  noStroke();  
  beginShape();  
  vertex(0, CANVAS_H);  
  for (let p of farPts) vertex(p.x, p.y);  
  vertex(WORLD_W, CANVAS_H);  
  endShape(CLOSE);  
  
  // Soft ridge rim-light — only during daylight  
  if (tod < 0.55) {  
    let ra = map(tod, 0, 0.55, 110, 0);  
    stroke(225, 212, 240, ra);  
    strokeWeight(1.5);  
    noFill();  
    beginShape();  
    for (let p of farPts) vertex(p.x, p.y - 2);  
    endShape();  
    noStroke();  
  }  
}  
  
// ======  
// MID HILLS — sage-green → dark silhouette
```

```
// =====

function drawMidHills() {
    let ns = lerp(1, 0.06, tod);
    let [r, g, b] = dn(162, 204, 170, ns);
    fill(r, g, b);
    noStroke();
    beginShape();
    vertex(0, CANVAS_H);
    for (let p of midPts) vertex(p.x, p.y);
    vertex(WORLD_W, CANVAS_H);
    endShape(CLOSE);
}

// =====

// NEAR GROUND — rich meadow base
// =====

function drawNearGround() {
    let ns = lerp(1, 0.06, tod);
    let [r, g, b] = dn(130, 182, 142, ns);
    fill(r, g, b);
    noStroke();
    beginShape();
    vertex(0, CANVAS_H);
    for (let p of gndPts) vertex(p.x, p.y);
    vertex(WORLD_W, CANVAS_H);
    endShape(CLOSE);
}
```

```
}
```

```
// =====  
// TREES — layered foliage, darken to silhouette at night  
// =====  
  
function drawTrees() {  
  
    const treeXs = [  
        85, 255, 435, 615, 800,  
        1015, 1195, 1385, 1570, 1755,  
        1945, 2135, 2325  
    ];  
  
  
    let ns = lerp(1, 0.055, tod);  
  
  
    for (let tx of treeXs) {  
        let gY = groundY(tx);  
        let h = 58 + noise(tx * 0.01) * 38;  
        let w = 46 + noise(tx * 0.02 + 5) * 18;  
  
        // Trunk  
        let [tr, tg, tb] = dn(148, 108, 78, ns);  
        fill(tr, tg, tb);  
        noStroke();  
        rect(tx - 5, gY - h + 12, 10, h);  
  
        // Three foliage blobs — overlapping gives soft depth
```

```

let [f1r, f1g, f1b] = dn(108, 162, 122, ns);
fill(f1r, f1g, f1b, 218);
ellipse(tx, gY - h - 4, w, w * 0.95);

let [f2r, f2g, f2b] = dn(92, 150, 110, ns);
fill(f2r, f2g, f2b, 200);
ellipse(tx - 14, gY - h + 9, w * 0.80, w * 0.80);

let [f3r, f3g, f3b] = dn(122, 175, 136, ns);
fill(f3r, f3g, f3b, 200);
ellipse(tx + 11, gY - h + 11, w * 0.74, w * 0.74);

}

}

// =====
// FLOWERS + GRASS — fade and desaturate toward night
// =====

function drawFlowers() {

let ns      = lerp(1, 0.055, tod);
let grassAlpha = lerp(162, 28, tod);

// Grass blades: two angled strokes per clump
strokeWeight(1.2);

for (let x = 0; x < WORLD_W; x += 18) {
let gy = groundY(x);
let bh = 7 + noise(x * 0.14) * 10;

```

```
let [gr, gg, gb] = dn(90, 152, 100, ns);

stroke(gr, gg, gb, grassAlpha);

line(x, gy, x - 2, gy - bh);

line(x + 7, gy, x + 9, gy - bh * 0.75);

}

noStroke();
```

```
const flowers = [

{x: 140, c: [255, 172, 185]},  

{x: 305, c: [255, 228, 142]},  

{x: 485, c: [212, 172, 255]},  

{x: 665, c: [255, 188, 200]},  

{x: 865, c: [255, 235, 152]},  

{x: 1055, c: [188, 172, 255]},  

{x: 1235, c: [255, 198, 168]},  

{x: 1425, c: [172, 218, 255]},  

{x: 1615, c: [255, 172, 212]},  

{x: 1805, c: [255, 225, 152]},  

{x: 1995, c: [192, 255, 192]},  

{x: 2195, c: [255, 192, 225]},  

{x: 2365, c: [255, 240, 158]},  

];
```

```
let flowerA = lerp(200, 18, tod);
```

```
for (let f of flowers) {
```

```
let gy = groundY(f.x);

// Stem

let [sr, sg, sb] = dn(92, 145, 80, ns);
stroke(sr, sg, sb);
strokeWeight(1.4);
line(f.x, gy, f.x, gy - 20);
noStroke();

// 6 petals in a ring

fill(f.c[0] * ns, f.c[1] * ns, f.c[2] * ns, flowerA);
for (let i = 0; i < 6; i++) {
  let a = (i / 6) * TWO_PI;
  ellipse(f.x + cos(a) * 6, (gy - 20) + sin(a) * 6, 8, 8);
}

// Bright center

fill(255 * ns, 242 * ns, 100 * ns);
ellipse(f.x, gy - 20, 7, 7);
}

}

=====

// PETALS / FIREFLIES

// Daytime: drifting pastel oval petals

// Night: glowing round fireflies (pale yellow-green)
```

```

// Morphs smoothly between the two as tod rises past 0.55
// =====

function drawPetals() {
    // nightFactor: 0 = full petal, 1 = full firefly

    let nightFactor = constrain(map(tod, 0.52, 0.88, 0, 1), 0, 1);

    noStroke();

    for (let p of petals) {
        // Update position

        p.wx += p.vx;

        p.y += sin(frameCount * 0.018 + p.ph) * 0.35;

        p.ang += p.aSpd;

        if (p.wx > WORLD_W) p.wx = 0;
        if (p.y > CANVAS_H - 25) p.y = 85;

        // Cull anything off-screen

        if (p.wx < camX - 15 || p.wx > camX + CANVAS_W + 15) continue;

        // Lerp colours: soft petal pink → firefly yellow-green

        let pr = lerp(p.r, 195, nightFactor);
        let pg = lerp(p.g, 255, nightFactor);
        let pb = lerp(p.b, 110, nightFactor);
        let pa = lerp(p.a, 215, nightFactor);

        // Firefly glow halo — grows in as night approaches
    }
}

```

```

if (nightFactor > 0.04) {

    fill(195, 255, 110, nightFactor * pa * 0.22);

    ellipse(p.wx, p.y, p.sz * 7, p.sz * 7);

}

// Shape: elongated petal by day → round dot by night

push();

translate(p.wx, p.y);

rotate(p.ang);

fill(pr, pg, pb, pa);

ellipse(0, 0,

lerp(p.sz * 2.3, p.sz * 1.15, nightFactor), // width

lerp(p.sz, p.sz * 1.1, nightFactor) // height

);

pop();

}

}

// =====

// SYMBOLS — 4 glowing discoveries that pulse on reveal

// =====

function drawSymbols() {

for (let s of symbols) {

s.ph += 0.055; // advance pulse continuously

}

// Visibility cull — world-space (inside translate(-camX))

```

```

let sx = s.wx - camX;
if (sx < -65 || sx > CANVAS_W + 65) continue;

let pulse = (sin(s.ph) + 1) * 0.5; // 0 → 1

// Radial glow: concentric rings fade outward
noStroke();
let maxR = 20 + pulse * 14;
for (let r = maxR; r > 0; r -= 2.5) {
  let a = map(r, maxR, 0, 0, 65 + pulse * 118);
  fill(255, 228, 115, a);
  ellipse(s.wx, s.wy, r * 2, r * 2);
}

// Symbol shape — alpha surges with pulse
let sa = 175 + pulse * 80;
fill(255, 245, 158, sa);
stroke(255, 205, 55, sa);
strokeWeight(1.5);

if (s.type === 'sun') drawSunSym(s.wx, s.wy, 10, pulse);
else if (s.type === 'leaf') drawLeafSym(s.wx, s.wy, 13, pulse);
else if (s.type === 'star') drawStarSym(s.wx, s.wy, 8, 18, 5);
else if (s.type === 'moon') drawMoonSym(s.wx, s.wy, 13, pulse);
}
}

```

```
// — Symbol: radiant sun with pulsing rays ——————  
  
function drawSunSym(x, y, r, pulse) {  
  
    noStroke();  
  
    fill(255, 235, 98, 200 + pulse * 55);  
  
    ellipse(x, y, r * 2, r * 2);  
  
  
    stroke(255, 215, 65, 175 + pulse * 80);  
  
    strokeWeight(2);  
  
    for (let i = 0; i < 8; i++) {  
  
        let a = (i / 8) * TWO_PI;  
  
        let i1 = r + 3;  
  
        let o1 = r + 8 + pulse * 5;  
  
        line(x + cos(a)*i1, y + sin(a)*i1, x + cos(a)*o1, y + sin(a)*o1);  
  
    }  
  
}
```

```
// — Symbol: botanical leaf with midrib vein ——————  
  
function drawLeafSym(x, y, r, pulse) {  
  
    fill(158, 225, 158, 200 + pulse * 55);  
  
    stroke(95, 182, 108, 220);  
  
    strokeWeight(1.2);  
  
    // Symmetrical leaf from mirrored bezier curves  
  
    beginShape();  
  
    vertex(x, y - r);  
  
    bezierVertex(x + r, y - r*0.3, x + r, y + r*0.3, x, y + r);
```

```
bezierVertex(x - r, y + r*0.3, x - r, y - r*0.3, x, y - r);
endShape(CLOSE);

// Midrib

stroke(95, 182, 108, 175);

line(x, y - r, x, y + r);

}
```

// — Symbol: 5-point star ——————

```
function drawStarSym(x, y, r1, r2, n) {

beginShape();

for (let i = 0; i < n * 2; i++) {

let r = (i % 2 === 0) ? r2 : r1;

let a = (i / (n * 2)) * TWO_PI - HALF_PI;

vertex(x + cos(a)*r, y + sin(a)*r);

}

endShape(CLOSE);

}
```

// — Symbol: crescent moon ——————

// Carved by overdrawing a smaller disc in the background colour.

```
function drawMoonSym(x, y, r, pulse) {

noStroke();

fill(255, 242, 185, 200 + pulse * 55);

ellipse(x, y, r * 2, r * 2);

// Carve out crescent with a deep-indigo cutout

fill(18, 13, 62, 240);
```

```

ellipse(x + r * 0.48, y, r * 1.52, r * 1.72);

}

// =====
// HELPER — ground surface Y at any world X
// =====

function groundY(wx) {
  return 316 + sin(wx * 0.006 + 1) * 9;
}

```

Here's a summary of everything that changed:

Symbol reorder & positioning

Sun → wx=150 — visible from the very first frames, sits left of screen center

Leaf → wx=950, wy = groundY(950) - 14 — bottom of the leaf symbol lands exactly on the ground surface

Star → wx=1500 — dusk area, sky already filling with stars when revealed

Moon → wx=2300 — enters the screen from the right near the end of the scroll (screen x ≈ 700 when camX=1600)

Crescent moon fix — the cutout disc now uses the night-sky indigo (18, 13, 62) instead of a fixed purple, so it blends correctly into the dark sky at night

Day-to-night transition with eased pacing

The computeTOD() function splits progress into three segments, each with their own smoothStep ease:

Segment	camX range	tod change	Feel
A	0 → 352	0 → 0.12	Slow — still clearly afternoon
B	352 → 832	0.12 → 0.88	Fast — accelerates around the Leaf at ~550
C	832 → 1600	0.88 → 1.0	Slow tail — eases into deep night

Visual effects that respond to tod

Sky gradient morphs through 4 color stops: lavender → orange → violet → navy

Sunset horizon glow band fades in and out using a sin() arch

Terrain + trees + flowers all scale to near-black silhouettes via dn() multiplier

Clouds fade out and shift to dark blue-grey wisps

Stars fade in from tod=0.32 with per-star twinkle

Petals morph to glowing fireflies from tod=0.52 onward — shape rounds out, colour shifts to yellow-green, a soft halo glow appears

add a start screen that tells users they can either autoscroll or play using left and right arrow.

add an autoscroll button and the ability to be turned on or off.

default should be left and right arrow controls.

Write sketch.js

716 lines

```
// =====
// MEDITATIVE NATURE SCROLL — AFTERNOON TO NIGHT
// Canvas: 800 × 400 | World: 2400 wide (3× canvas)
// Narrative: camera drifts left→right through the day.
// Symbols in discovery order: Sun → Leaf → Star → Moon
// Sun (wx=150) — left side of screen, full afternoon
```

```
// Leaf (wx=950) — resting on the ground, golden hour  
// Star (wx=1500) — dusk sky, stars emerging  
// Moon (wx=2300) — right side of screen, deep night  
// p5.js only — no external libraries  
// =====
```

```
// — WORLD & CAMERA ——————
```

```
const WORLD_W = 2400;  
const CANVAS_W = 800;  
const CANVAS_H = 400;  
const SPEED = 0.5; // autoscroll drift speed (px/frame)  
const ARROW_SPD = 2.2; // manual arrow-key speed (px/frame)
```

```
let camX = 0;
```

```
// — TIME OF DAY ——————
```

```
// tod: 0 = bright afternoon, 1 = deep night  
let tod = 0;
```

```
// — UI STATE ——————
```

```
// gameState: 'start' shows the intro card; 'playing' runs the world  
let gameState = 'start';  
let autoScroll = false; // default control is left/right arrows
```

```
// Autoscroll toggle button — bottom-right corner
```

```
const BTN_W = 152;
```

```
const BTN_H = 24;  
  
const BTN_X = CANVAS_W - BTN_W - 8;  
  
const BTN_Y = CANVAS_H - BTN_H - 8;  
  
  
// — TERRAIN PROFILES ——————  
  
let farPts = [];  
  
let midPts = [];  
  
let gndPts = [];  
  
  
// — FLOATING PETALS / FIREFLIES ——————  
  
let petals = [];  
  
  
// — STARS (screen-space) ——————  
  
let stars = [];  
  
  
// — HIDDEN SYMBOLS ——————  
  
let symbols = [];  
  
  
// ======  
  
// SETUP  
  
// ======  
  
function setup() {  
  
  createCanvas(CANVAS_W, CANVAS_H);  
  
  
  // Terrain from overlapping sine waves  
  
  for (let x = 0; x <= WORLD_W; x += 6) {
```

```
farPts.push(createVector(x,
 212 + sin(x * 0.003) * 52
 + sin(x * 0.007 + 1.2) * 26
));
midPts.push(createVector(x,
 270 + sin(x * 0.0045 + 2) * 34
 + sin(x * 0.009 + 4.5) * 15
));
gndPts.push(createVector(x,
 316 + sin(x * 0.006 + 1) * 9
));
}
}
```

```
// Floating petals scattered across the world
for (let i = 0; i < 55; i++) {
  petals.push({
    wx: random(WORLD_W),
    y: random(80, 345),
    sz: random(3, 7.5),
    vx: random(0.18, 0.65),
    ph: random(TWO_PI),
    ang: random(TWO_PI),
    aSpd: random(-0.025, 0.025),
    r: random(230, 255),
    g: random(148, 218),
    b: random(182, 234),
  });
}
```

```

    a: random(140, 210)
  });
}

// Stars in screen-space (sky is infinite — no parallax)

for (let i = 0; i < 90; i++) {
  stars.push({
    x: random(CANVAS_W),
    y: random(CANVAS_H * 0.62),
    sz: random(0.8, 2.4),
    ph: random(TWO_PI)
  });
}

// Hidden symbols: Sun → Leaf → Star → Moon

let leafGY = 316 + sin(950 * 0.006 + 1) * 9; // groundY(950) ≈ 320

symbols = [
  { wx: 150, wy: 158, type: 'sun', ph: 0 },
  { wx: 950, wy: leafGY - 14, type: 'leaf', ph: HALF_PI },
  { wx: 1500, wy: 188, type: 'star', ph: PI },
  { wx: 2300, wy: 162, type: 'moon', ph: PI+HALF_PI }
];

}

// =====

```

```
// DRAW
// =====

function draw() {
    // During the start screen the world is visible but frozen.
    // tod stays 0 (afternoon) so it looks inviting.

    if (gameState === 'playing') {
        updateCamera();
        tod = computeTOD();
    } else {
        tod = 0;
    }

    // — Draw world layers (always rendered, even on start screen) —
    drawSky();
    drawStars();

    push(); translate(-camX * 0.15, 0); drawClouds(); pop();
    push(); translate(-camX * 0.35, 0); drawFarHills(); pop();
    push(); translate(-camX * 0.62, 0); drawMidHills(); pop();

    push();
    translate(-camX, 0);
    drawNearGround();
    drawTrees();
    drawFlowers();
    drawPetals();
```

```
drawSymbols();

pop();

// — UI overlay ——————
if (gameState === 'start') {
    drawStartScreen();
} else {
    drawAutoScrollBtn();
}

}

}

// =====
// CAMERA UPDATE — called only while playing
// =====

function updateCamera() {
    if (autoScroll) {
        // Automatic drift — loops seamlessly
        camX = (camX + SPEED) % (WORLD_W - CANVAS_W);
    } else {
        // Manual arrow-key control with clamped bounds
        if (keyIsDown(LEFT_ARROW)) camX = max(0, camX - ARROW_SPD);
        if (keyIsDown(RIGHT_ARROW)) camX = min(WORLD_W - CANVAS_W, camX + ARROW_SPD);
    }
}
```

```
// =====  
  
// MOUSE PRESSED — start screen click, button toggle  
// =====  
  
function mousePressed() {  
  
    if (gameState === 'start') {  
  
        gameState = 'playing';  
  
        return;  
  
    }  
  
    // Autoscroll button hit test  
  
    if (mouseX >= BTN_X && mouseX <= BTN_X + BTN_W &&  
        mouseY >= BTN_Y && mouseY <= BTN_Y + BTN_H) {  
  
        autoScroll = !autoScroll;  
  
    }  
  
}  
  
// =====  
  
// KEY PRESSED — any key dismisses the start screen  
// =====  
  
function keyPressed() {  
  
    if (gameState === 'start') {  
  
        gameState = 'playing';  
  
    }  
  
}  
  
// =====  
  
// START SCREEN — warm card overlay on top of frozen world
```

```
// ======

function drawStartScreen() {

    // Soft dark vignette behind the card

    noStroke();

    fill(18, 12, 38, 155);

    rect(0, 0, CANVAS_W, CANVAS_H);


    let cx = CANVAS_W / 2;

    let cy = CANVAS_H / 2 - 4;

    let cw = 470;

    let ch = 205;


    // Card drop-shadow

    fill(0, 0, 0, 55);

    rect(cx - cw/2 + 5, cy - ch/2 + 5, cw, ch, 16);


    // Card face — warm parchment

    fill(255, 248, 232, 242);

    rect(cx - cw/2, cy - ch/2, cw, ch, 14);


    // Card border accent

    stroke(215, 185, 145, 180);

    strokeWeight(1.5);

    noFill();

    rect(cx - cw/2, cy - ch/2, cw, ch, 14);

    noStroke();
```

```
// — Title ——————  
textAlign(CENTER, CENTER);  
textSize(17);  
fill(72, 50, 28);  
text('Meditative Nature Scroll', cx, cy - ch/2 + 28);  
  
// Divider  
stroke(205, 178, 140, 160);  
strokeWeight(1);  
line(cx - cw/2 + 32, cy - ch/2 + 50, cx + cw/2 - 32, cy - ch/2 + 50);  
noStroke();  
  
// — Control rows ——————  
textSize(13);  
let rowY1 = cy - 22;  
let rowY2 = cy + 12;  
let iconX = cx - 175;  
let descX = cx - 132;  
  
// Row 1 — Arrow keys  
fill(60, 42, 22);  
textAlign(LEFT, CENTER);  
text('\u2190 \u2192', iconX, rowY1);      // ← →  
fill(105, 78, 50);  
text('Arrow keys \u2014 explore at your own pace', descX, rowY1);
```

```

// Row 2 — Autoscroll

fill(60, 42, 22);

text('\u21BB', iconX, rowY2);           // ⏪

fill(105, 78, 50);

text('Autoscroll \u2014 camera drifts on its own', descX, rowY2);

// Sub-hint

textSize(11);

fill(148, 115, 78);

textAlign(CENTER, CENTER);

text('Toggle autoscroll with the button in the bottom-right corner', cx, cy + ch/2 - 38);

// — Start prompt — pulse gently ——————
let pulse = (sin(frameCount * 0.07) + 1) * 0.5;

fill(88, 62, 34, lerp(155, 245, pulse));

textSize(12);

text('Click anywhere or press any key to begin', cx, cy + ch/2 - 18);

}

// =====
// AUTOSCROLL BUTTON — bottom-right during play
// =====

function drawAutoScrollBtn() {

let x = BTN_X;

let y = BTN_Y;

```

```
// Drop shadow
noStroke();
fill(0, 0, 0, 45);
rect(x + 2, y + 2, BTN_W, BTN_H, 7);

// Face — green tint when on, warm cream when off
if (autoScroll){
    fill(88, 168, 112, 218);
} else {
    fill(242, 228, 208, 218);
}
rect(x, y, BTN_W, BTN_H, 7);

// Border
if (autoScroll){
    stroke(60, 132, 82, 200);
} else {
    stroke(185, 158, 122, 190);
}
strokeWeight(1);
noFill();
rect(x, y, BTN_W, BTN_H, 7);
noStroke();

// Label
```

```

textAlign(CENTER, CENTER);
textSize(11);
if (autoScroll) {
  fill(22, 55, 32);
  text('\u21BB AUTOSCROLL: ON', x + BTN_W/2, y + BTN_H/2);
} else {
  fill(75, 52, 28);
  text('\u2190\u2192 ARROWS: ON', x + BTN_W/2, y + BTN_H/2);
}
}

// =====
// HELPERS
// =====

// Classic smooth-step: ease in AND out over [0,1]
function smoothStep(t){
  t = constrain(t, 0, 1);
  return t * t * (3 - 2 * t);
}

// Compute time-of-day from camX.
// Three segments give a slow → FAST → slow easing curve:
// Segment A camX 0→352 (p 0→0.22) tod 0→0.12 slow afternoon
// Segment B camX 352→832 (p 0.22→0.52) tod 0.12→0.88 FAST at the Leaf
// Segment C camX 832→1600 (p 0.52→1) tod 0.88→1.0 slow tail to night

```

```

function computeTOD() {
    let p = camX / (WORLD_W - CANVAS_W);

    if (p < 0.22) {
        return lerp(0, 0.12, smoothStep(p / 0.22));
    } else if (p < 0.52) {
        return lerp(0.12, 0.88, smoothStep((p - 0.22) / 0.30));
    } else {
        return lerp(0.88, 1.0, smoothStep((p - 0.52) / 0.48));
    }
}

// Interpolate through an array of [r,g,b] color stops
function lerpStops(stops, t) {
    let n = stops.length - 1;
    let i = min(floor(t * n), n - 1);
    let lt = (t * n) - i;
    return [
        lerp(stops[i][0], stops[i+1][0], lt),
        lerp(stops[i][1], stops[i+1][1], lt),
        lerp(stops[i][2], stops[i+1][2], lt)
    ];
}

// Ground surface Y at any world X (matches gndPts formula)
function groundY(wx) {

```

```
return 316 + sin(wx * 0.006 + 1) * 9;  
}  
  
  
// Scale an RGB colour toward black for night silhouette  
  
function dn(r, g, b, scale){  
    return [r * scale, g * scale, b * scale];  
}  
  
  
// ======  
  
// SKY — afternoon → golden hour → dusk → night  
  
// ======  
  
function drawSky() {  
    noStroke();  
  
  
    const topStops = [  
        [185, 172, 230], // afternoon: soft lavender-blue  
        [255, 152, 75], // golden hr: warm orange  
        [72, 42, 138], // dusk: deep violet  
        [8, 12, 55] // night: navy  
    ];  
  
    const botStops = [  
        [255, 210, 178], // afternoon: peachy cream  
        [255, 90, 30], // golden hr: red-orange  
        [178, 68, 115], // dusk: magenta-rose  
        [18, 13, 62] // night: dark indigo  
    ];
```

```

let topC = lerpStops(topStops, tod);
let botC = lerpStops(botStops, tod);

for (let y = 0; y < CANVAS_H; y++) {
    let t = y / CANVAS_H;
    stroke(
        lerp(topC[0], botC[0], t),
        lerp(topC[1], botC[1], t),
        lerp(topC[2], botC[2], t)
    );
    line(0, y, CANVAS_W, y);
}

noStroke();

// Sunset horizon glow — warm band, sine-arched so it fades in and out
if (tod > 0.14 && tod < 0.74) {
    let intensity = sin(map(tod, 0.14, 0.74, 0, PI));
    let horizY  = CANVAS_H * 0.52;
    for (let y = CANVAS_H * 0.36; y < CANVAS_H * 0.68; y++) {
        let dy = abs(y - horizY) / (CANVAS_H * 0.16);
        let a  = max(0, 1 - dy) * intensity * 95;
        stroke(255, 145, 38, a);
        line(0, y, CANVAS_W, y);
    }
    noStroke();
}

```

```
}

}

// =====

// STARS — screen-space, twinkle in at dusk

// =====

function drawStars() {

  if (tod < 0.32) return;

  let alpha = map(tod, 0.32, 0.72, 0, 255);

  noStroke();

  for (let s of stars) {

    let tw = (sin(frameCount * 0.038 + s.ph) + 1) * 0.5;

    fill(245, 248, 255, alpha * (0.62 + tw * 0.38));

    ellipse(s.x, s.y, s.sz + tw * 0.55);

  }

}

// =====

// CLOUDS — fade and tint toward dark wisps at night

// =====

function drawClouds() {

  let baseA = lerp(192, 28, tod);

  let cr  = lerp(255, 48, tod);

  let cg  = lerp(245, 42, tod);

  let cb  = lerp(250, 88, tod);

}
```

```

const clouds = [
  {x: 150, y: 65, s: 1.2 },
  {x: 490, y: 52, s: 0.9 },
  {x: 830, y: 70, s: 1.4 },
  {x: 1170, y: 56, s: 1.0 },
  {x: 1500, y: 68, s: 1.15 },
  {x: 1840, y: 50, s: 0.85 },
  {x: 2180, y: 73, s: 1.1 },
];

noStroke();

for (let c of clouds) {
  fill(cr, cg, cb, baseA);
  ellipse(c.x,      c.y,      80*c.s, 38*c.s);
  ellipse(c.x - 34*c.s, c.y + 10*c.s, 56*c.s, 30*c.s);
  ellipse(c.x + 38*c.s, c.y + 8*c.s, 60*c.s, 28*c.s);
  let blushA = max(0, lerp(52, -10, tod));
  fill(255, 220, 235, blushA);
  ellipse(c.x, c.y - 5, 60*c.s, 22*c.s);
}

// =====
// FAR HILLS — lavender → black silhouette
// =====

function drawFarHills() {

```

```

let ns = lerp(1, 0.07, tod);

let [r, g, b] = dn(205, 188, 225, ns);

fill(r, g, b);

noStroke();

beginShape();

vertex(0, CANVAS_H);

for (let p of farPts) vertex(p.x, p.y);

vertex(WORLD_W, CANVAS_H);

endShape(CLOSE);

if (tod < 0.55) {

let ra = map(tod, 0, 0.55, 110, 0);

stroke(225, 212, 240, ra);

strokeWeight(1.5);

noFill();

beginShape();

for (let p of farPts) vertex(p.x, p.y - 2);

endShape();

noStroke();

}

}

// =====

// MID HILLS — sage-green → silhouette

// =====

function drawMidHills() {

```

```
let ns = lerp(1, 0.06, tod);

let [r, g, b] = dn(162, 204, 170, ns);

fill(r, g, b);

noStroke();

beginShape();

vertex(0, CANVAS_H);

for (let p of midPts) vertex(p.x, p.y);

vertex(WORLD_W, CANVAS_H);

endShape(CLOSE);

}

// =====

// NEAR GROUND

// =====

function drawNearGround() {

let ns = lerp(1, 0.06, tod);

let [r, g, b] = dn(130, 182, 142, ns);

fill(r, g, b);

noStroke();

beginShape();

vertex(0, CANVAS_H);

for (let p of gndPts) vertex(p.x, p.y);

vertex(WORLD_W, CANVAS_H);

endShape(CLOSE);

}
```

```
// =====
// TREES — darken to silhouette at night
// =====

function drawTrees() {

  const treeXs = [
    85, 255, 435, 615, 800,
    1015, 1195, 1385, 1570, 1755,
    1945, 2135, 2325
  ];

  let ns = lerp(1, 0.055, tod);

  for (let tx of treeXs) {

    let gY = groundY(tx);
    let h = 58 + noise(tx * 0.01) * 38;
    let w = 46 + noise(tx * 0.02 + 5) * 18;

    let [tr, tg, tb] = dn(148, 108, 78, ns);
    fill(tr, tg, tb); noStroke();
    rect(tx - 5, gY - h + 12, 10, h);

    let [f1r, f1g, f1b] = dn(108, 162, 122, ns);
    fill(f1r, f1g, f1b, 218);
    ellipse(tx, gY - h - 4, w, w * 0.95);

    let [f2r, f2g, f2b] = dn(92, 150, 110, ns);
    fill(f2r, f2g, f2b, 200);
  }
}
```

```

ellipse(tx - 14, gY - h + 9, w * 0.80, w * 0.80);

let [f3r, f3g, f3b] = dn(122, 175, 136, ns);
fill(f3r, f3g, f3b, 200);
ellipse(tx + 11, gY - h + 11, w * 0.74, w * 0.74);

}

}

// =====

// FLOWERS + GRASS — fade and desaturate at night
// =====

function drawFlowers() {

let ns      = lerp(1, 0.055, tod);
let grassAlpha = lerp(162, 28, tod);

strokeWeight(1.2);

for (let x = 0; x < WORLD_W; x += 18) {

let gy = groundY(x);
let bh = 7 + noise(x * 0.14) * 10;
let [gr, gg, gb] = dn(90, 152, 100, ns);
stroke(gr, gg, gb, grassAlpha);
line(x, gy, x - 2, gy - bh);
line(x + 7, gy, x + 9, gy - bh * 0.75);

}
noStroke();
}

```

```
const flowers = [
  { x: 140, c: [255, 172, 185] },
  { x: 305, c: [255, 228, 142] },
  { x: 485, c: [212, 172, 255] },
  { x: 665, c: [255, 188, 200] },
  { x: 865, c: [255, 235, 152] },
  { x: 1055, c: [188, 172, 255] },
  { x: 1235, c: [255, 198, 168] },
  { x: 1425, c: [172, 218, 255] },
  { x: 1615, c: [255, 172, 212] },
  { x: 1805, c: [255, 225, 152] },
  { x: 1995, c: [192, 255, 192] },
  { x: 2195, c: [255, 192, 225] },
  { x: 2365, c: [255, 240, 158] },
];
```

```
let flowerA = lerp(200, 18, tod);

for (let f of flowers) {
  let gy = groundY(f.x);
  let [sr, sg, sb] = dn(92, 145, 80, ns);
  stroke(sr, sg, sb);
  strokeWeight(1.4);
  line(f.x, gy, f.x, gy - 20);
  noStroke();
```

```

fill(f.c[0] * ns, f.c[1] * ns, f.c[2] * ns, flowerA);

for (let i = 0; i < 6; i++) {
  let a = (i / 6) * TWO_PI;
  ellipse(f.x + cos(a) * 6, (gy - 20) + sin(a) * 6, 8, 8);
}

fill(255 * ns, 242 * ns, 100 * ns);
ellipse(f.x, gy - 20, 7, 7);

}

}

// =====
// PETALS / FIREFLIES

// Daytime: drifting pastel oval petals
// Night: glowing round fireflies (pale yellow-green)
// =====

function drawPetals() {

let nightFactor = constrain(map(tod, 0.52, 0.88, 0, 1), 0, 1);

noStroke();

for (let p of petals) {

  p.wx += p.vx;

  p.y += sin(frameCount * 0.018 + p.ph) * 0.35;

  p.ang += p.aSpd;

  if (p.wx > WORLD_W) p.wx = 0;

  if (p.y > CANVAS_H - 25) p.y = 85;
}

```

```

if (p.wx < camX - 15 || p.wx > camX + CANVAS_W + 15) continue;

let pr = lerp(p.r, 195, nightFactor);
let pg = lerp(p.g, 255, nightFactor);
let pb = lerp(p.b, 110, nightFactor);
let pa = lerp(p.a, 215, nightFactor);

if (nightFactor > 0.04) {
    fill(195, 255, 110, nightFactor * pa * 0.22);
    ellipse(p.wx, p.y, p.sz * 7, p.sz * 7);
}

push();
translate(p.wx, p.y);
rotate(p.ang);
fill(pr, pg, pb, pa);
ellipse(0, 0,
    lerp(p.sz * 2.3, p.sz * 1.15, nightFactor),
    lerp(p.sz, p.sz * 1.1, nightFactor)
);
pop();
}

}

// =====
// SYMBOLS — glowing discoveries; pulse when revealed

```

```
// =====

function drawSymbols() {
    for (let s of symbols) {
        s.ph += 0.055;

        let sx = s.wx - camX;
        if (sx < -65 || sx > CANVAS_W + 65) continue;

        let pulse = (sin(s.ph) + 1) * 0.5;

        noStroke();
        let maxR = 20 + pulse * 14;
        for (let r = maxR; r > 0; r -= 2.5) {
            let a = map(r, maxR, 0, 0, 65 + pulse * 118);
            fill(255, 228, 115, a);
            ellipse(s.wx, s.wy, r * 2, r * 2);
        }

        let sa = 175 + pulse * 80;
        fill(255, 245, 158, sa);
        stroke(255, 205, 55, sa);
        strokeWeight(1.5);

        if (s.type === 'sun') drawSunSym(s.wx, s.wy, 10, pulse);
        else if (s.type === 'leaf') drawLeafSym(s.wx, s.wy, 13, pulse);
        else if (s.type === 'star') drawStarSym(s.wx, s.wy, 8, 18, 5);
    }
}
```

```
    else if (s.type === 'moon') drawMoonSym(s.wx, s.wy, 13, pulse);  
}  
}  
  
// — Symbol: radiant sun —
```

```
function drawSunSym(x, y, r, pulse) {  
    noStroke();  
    fill(255, 235, 98, 200 + pulse * 55);  
    ellipse(x, y, r * 2, r * 2);  
    stroke(255, 215, 65, 175 + pulse * 80);  
    strokeWeight(2);  
    for (let i = 0; i < 8; i++) {  
        let a = (i / 8) * TWO_PI;  
        let i1 = r + 3;  
        let o1 = r + 8 + pulse * 5;  
        line(x + cos(a)*i1, y + sin(a)*i1, x + cos(a)*o1, y + sin(a)*o1);  
    }  
}
```

```
// — Symbol: botanical leaf —
```

```
function drawLeafSym(x, y, r, pulse) {  
    fill(158, 225, 158, 200 + pulse * 55);  
    stroke(95, 182, 108, 220);  
    strokeWeight(1.2);  
    beginShape();  
    vertex(x, y - r);
```

```
bezierVertex(x + r, y - r*0.3, x + r, y + r*0.3, x, y + r);
bezierVertex(x - r, y + r*0.3, x - r, y - r*0.3, x, y - r);
endShape(CLOSE);
stroke(95, 182, 108, 175);
line(x, y - r, x, y + r);
}
```

// — Symbol: 5-point star —————

```
function drawStarSym(x, y, r1, r2, n) {
beginShape();
for (let i = 0; i < n * 2; i++) {
let r = (i % 2 === 0) ? r2 : r1;
let a = (i / (n * 2)) * TWO_PI - HALF_PI;
vertex(x + cos(a)*r, y + sin(a)*r);
}
endShape(CLOSE);
}
```

// — Symbol: crescent moon —————

```
function drawMoonSym(x, y, r, pulse) {
noStroke();
fill(255, 242, 185, 200 + pulse * 55);
ellipse(x, y, r * 2, r * 2);
// Carve crescent with a disc matching the night sky
fill(18, 13, 62, 240);
ellipse(x + r * 0.48, y, r * 1.52, r * 1.72);
```

```
}

// =====
// HELPER — ground surface Y at any world X
// =====

function groundY(wx) {
    return 316 + sin(wx * 0.006 + 1) * 9;
}
```

Here's what was added and how each piece works:

Start screen

Draws the world frozen at tod=0 (bright afternoon) as a live background — petals still drift through it

A warm parchment card centered on canvas explains both control modes with Unicode arrow/cycle icons

The "Click or press any key to begin" prompt pulses gently using a sin() fade

Any click or keypress switches gameState from 'start' → 'playing'

Autoscroll toggle button

Positioned bottom-right with an 8px margin (BTN_X, BTN_Y)

Off (default): warm cream background, shows ←→ ARROWS: ON

On: soft green background, shows ⌂ AUTOSCROLL: ON

Click detection uses a simple bounding box in mousePressed()

Default — arrow key control

autoScroll = false out of the gate

`keyIsDown(LEFT_ARROW) / keyIsDown(RIGHT_ARROW)` run every frame in `updateCamera()` at `ARROW_SPD = 2.2 px/frame` (about 4× autoscroll speed so it feels responsive)

Camera is clamped with `max(0, ...)` and `min(WORLD_W - CANVAS_W, ...)` so you can't scroll past the world edges

Autoscroll uses the original `SPEED = 0.5` and loops seamlessly