Process & Decision Documentation

Project/Assignment Decisions

I chose to redesign the blob's movement and environment to express fear rather than adding enemies or explicit threats. I focused on subtle instability in movement, faster "breathing," and reduced visibility in dark areas to communicate anxiety while keeping the game playable. This decision allowed fear to emerge through interaction and atmosphere rather than punishment or difficulty spikes.

Role-Based Process Evidence

Name: Josh Metivier
Role(s): Designer, Programmer
Primary responsibility for this work: Gameplay behaviour, emotional expression through movement and environment

Goal of Work Session

To redesign an existing p5.js blob platformer so that the blob's movement and the surrounding environment communicate a feeling of fear.

Tools, Resources, or Inputs Used

- p5.js platformer starter code
- Lecture materials for Side Quests
- ChatGPT (GenAI) for brainstorming and debugging
- Iterative testing in browser (Chrome)

GenAI Documentation

Date Used: January 25, 2026 **Tool**
Disclosure: ChatGPT 5.2

Purpose of Use:
To brainstorm ways of expressing fear through movement and environment, and to assist with debugging and refining p5.js code.
Summary of Interaction:
ChatGPT was used to generate design ideas for fear-based movement and environmental cues, and to help diagnose and fix platforming bugs during implementation.

Human Decision Point(s):
I selected a design approach focused on anxious movement and lighting rather than enemies or damage. I modified and simplified suggested mechanics to maintain playability.

Integrity & Verification Note:

All GenAI suggestions were reviewed, tested, and adjusted manually. Final decisions about mechanics and tuning were made through playtesting.

Scope of GenAI Use:
GenAI supported ideation and debugging but did not independently complete the assignment.

Limitations or Misfires:
Some suggested mechanics introduced unintended bugs or reduced playability and were revised or removed.

Summary of Process (Human + Tool)

I began by identifying fear as an emotion best expressed through instability rather than speed or power. I iteratively implemented movement jitter, panic-based animation changes, and environmental lighting, testing each change to ensure the platforming remained functional. Bugs related to collision and platform interaction were resolved through incremental debugging and refinement.

Decision Points & Trade-oEs

I chose not to include enemies or failure states in order to focus on emotional expression through movement and atmosphere. This trade-off reduced complexity while keeping the redesign aligned with the assignment’s goals.

Verification & Judgement

I evaluated the effectiveness of the redesign through repeated playtesting, focusing on whether fear was communicated without making the game frustrating. The final result consistently conveyed anxiety through motion and lighting while remaining playable.

Limitations, Dead Ends, or Open Questions

Additional elements such as sound design or enemy behaviour could further enhance fear, but these were excluded due to time and scope constraints.

Appendix

User Prompt:

I have this code for a blob that you can see here at the end of the prompt. Redesign the blob’s movement and environment to express fear. Give me some ideas for this that we can brainstorm and then we'll mess around with the code to see what works best.

Summary of GenAI Response:
ChatGPT proposed multiple design directions for expressing fear through movement and environment, including jittery motion, skittish acceleration, trembling while idle, reduced visibility through darkness, and environmental cues such as shadows. These ideas were grouped into conceptual "packages," from which a direction focused on anxious movement and lighting ("Package A") was selected.

Design Direction Selection

User Prompt:

i want to do package A.

Summary of GenAI Response:
ChatGPT outlined how "Package A" could be implemented using a panic variable affecting movement stability, animation speed, and environmental lighting, while maintaining player control and avoiding enemies.

Implementation & Debugging – Hitbox and Visual Issues

User Prompt:

the blob is huge and there isnt really even dark or light areas, its all just the same. The hitbox for the blob is also smaller than the blob so the majority of it can go through the platforms

Summary of GenAI Response:
ChatGPT identified that the blob's visual wobble exceeded its collision radius and that lighting cues were not clearly visible. Suggested fixes included visually rendering light pools and adjusting collision logic so the hitbox better matched the perceived size of the blob.

Implementation & Debugging – Hitbox Too Large

User Prompt:

everything is looking real good right now. The issue i have found is that the hitbox for the blob is now bigger than the blob itself

Summary of GenAI Response:
ChatGPT explained that using the maximum wobble radius for collision caused an oversized hitbox. A revised collision radius based on an averaged visual size was suggested, along with an optional debug toggle to visualize the hitbox.

Implementation & Debugging – Teleporting / Snapping Bug

User Prompt:

When the blob goes a certain distance over to the right, it glitches and shoots back to the left corner where it started.

Summary of GenAI Response:
ChatGPT identified that the floor platform was being treated as a horizontal collider, causing unintended collision resolution. The fix involved excluding the floor from horizontal collision checks and adding a safety clamp after vertical collision resolution.

Implementation & Debugging – Platform Landing Issues

User Prompt:

there is a glitch with jumping on the platform… i can jump on the rightmost platform but not the leftmost one

Summary of GenAI Response:
ChatGPT diagnosed the issue as instability caused by dynamically changing hitbox size and fearinduced jitter. The solution involved stabilizing the physics hitbox and limiting tremble effects while grounded to preserve platforming reliability.

Human Role in Decision-Making

All GenAI suggestions were selectively implemented, modified, or rejected based on manual testing and alignment with assignment goals. Final tuning decisions regarding collision behavior, movement feel, and environmental visibility were made by the student through iterative playtesting.

Full Gen AI Transcript

I have this code for a blob that you can see here at the end of the prompt. Redesign the blob's movement and environment to express fear. Give me some ideas for this that we can brainstorm and then we'll mess around with the code to see what works best. // Y-position of the floor (ground level) let floorY3; // Player character (soft, animated blob) let blob3 = { // Position (centre of the blob) x: 80, y: 0, // Visual properties r: 26, // Base radius points: 48, // Number of points used to draw the blob wobble: 7, // Edge deformation amount wobbleFreq: 0.9, // Time values for breathing animation t: 0, tSpeed: 0.01, // Physics: velocity vx: 0, // Horizontal velocity

```
vy: 0, // Vertical velocity // Movement tuning accel: 0.55, // Horizontal acceleration maxRun: 4.0,
// Maximum horizontal speed gravity: 0.65, // Downward force jumpV: -11.0, // Initial jump
impulse // State onGround: false, // True when standing on a platform // Friction frictionAir:
0.995, // Light friction in air frictionGround: 0.88, // Stronger friction on ground }; // List of solid
platforms the blob can stand on // Each platform is an axis-aligned rectangle (AABB) let
platforms = []; function setup() { createCanvas(640, 360); // Define the floor height floorY3 =
height - 36; noStroke(); textFont("sans-serif"); textSize(14); // Create platforms (floor + steps)
platforms = [ { x: 0, y: floorY3, w: width, h: height - floorY3 }, // floor { x: 120, y: floorY3 - 70,
w: 120, h: 12 }, // low step { x: 300, y: floorY3 - 120, w: 90, h: 12 }, // mid step { x: 440, y:
floorY3 - 180, w: 130, h: 12 }, // high step { x: 520, y: floorY3 - 70, w: 90, h: 12 }, // return ramp
]; // Start the blob resting on the floor blob3.y = floorY3 - blob3.r - 1; } function draw() {
background(240); // --- Draw all platforms --- fill(200); for (const p of platforms) { rect(p.x, p.y,
p.w, p.h); } // --- Input: left/right movement --- let move = 0; if (keyIsDown(65) ||
keyIsDown(LEFT_ARROW)) move -= 1; // A or ← if (keyIsDown(68) ||
keyIsDown(RIGHT_ARROW)) move += 1; // D or → blob3.vx += blob3.accel * move; // ---
Apply friction and clamp speed --- blob3.vx *= blob3.onGround ? blob3.frictionGround :
blob3.frictionAir; blob3.vx = constrain(blob3.vx, -blob3.maxRun, blob3.maxRun); // --- Apply
gravity --- blob3.vy += blob3.gravity; // --- Collision representation --- // We collide using a
rectangle (AABB), // even though the blob is drawn as a circle let box = { x: blob3.x - blob3.r, y:
blob3.y - blob3.r, w: blob3.r * 2, h: blob3.r * 2, }; // --- STEP 1: Move horizontally, then resolve
X collisions --- box.x += blob3.vx; for (const s of platforms) { if (overlap(box, s)) { if (blob3.vx
> 0) { // Moving right → hit the left side of a platform box.x = s.x - box.w; } else if (blob3.vx <
0) { // Moving left → hit the right side of a platform box.x = s.x + s.w; } blob3.vx = 0; } } // ---
STEP 2: Move vertically, then resolve Y collisions --- box.y += blob3.vy; blob3.onGround =
false; for (const s of platforms) { if (overlap(box, s)) { if (blob3.vy > 0) { // Falling → land on top
of a platform box.y = s.y - box.h; blob3.vy = 0; blob3.onGround = true; } else if (blob3.vy < 0) {
// Rising → hit the underside of a platform box.y = s.y + s.h; blob3.vy = 0; } } } // --- Convert
collision box back to blob centre --- blob3.x = box.x + box.w / 2; blob3.y = box.y + box.h / 2; //
Keep blob inside the canvas horizontally blob3.x = constrain(blob3.x, blob3.r, width - blob3.r); //
--- Draw the animated blob --- blob3.t += blob3.tSpeed; drawBlobCircle(blob3); // --- HUD ---
fill(0); text("Move: A/D or ←/→ • Jump: Space/W/↑ • Land on platforms", 10, 18); } // Axis-
Aligned Bounding Box (AABB) overlap test // Returns true if rectangles a and b intersect
function overlap(a, b) { return ( a.x < b.x + b.w && a.x + a.w > b.x && a.y < b.y + b.h && a.y +
a.h > b.y ); } // Draws the blob using Perlin noise for a soft, breathing effect function
drawBlobCircle(b) { fill(20, 120, 255); beginShape(); for (let i = 0; i < b.points; i++) { const a =
(i / b.points) * TAU; // Noise-based radius offset const n = noise( cos(a) * b.wobbleFreq + 100,
sin(a) * b.wobbleFreq + 100, b.t, ); const r = b.r + map(n, 0, 1, -b.wobble, b.wobble); vertex(b.x +
cos(a) * r, b.y + sin(a) * r); } endShape(CLOSE); } // Jump input (only allowed when grounded)
function keyPressed() { if ( (key === " " || key === "W" || key === "w" || keyCode ===
UP_ARROW) && blob3.onGround ) { blob3.vy = blob3.jumpV; blob3.onGround = false; } } /*
In-class tweaks for experimentation: • Add a new platform: platforms.push({ x: 220, y: floorY3 -
150, w: 80, h: 12 }); • “Ice” feel → frictionGround = 0.95 • “Sand” feel → frictionGround = 0.80
*/
```

Awesome — you've already got a "cute, stable platformer blob." To make it *feel fear*, you want the movement to get a little unreliable, reactive, and defensive… and the world to feel hostile/uncertain. Here are a bunch of brainstormable directions (mix + match).

Movement ideas that read as fear

1) Tremble / panic jitter (but still controllable)

- When standing still, the blob *can't fully stop*: tiny side-to-side micro-movement + subtle vertical shake.
- Wobble increases when near danger or in darkness.
- Visual: wobbleFreq speeds up slightly (like rapid breathing).

Knobs: "trembleStrength", "panicLevel", "panicDecay".

---

2) Skittish acceleration (overreacts, then regrets)

- First frame of input gives a **bigger-than-normal** shove (like flinching).
- After that, friction kicks harder to "freeze up."
- So movement feels like: *jerk → stall → jerk*.

Reads as: anxious, hesitant.

---

3) Freeze response (fear paralysis)

- When a threat gets close or a loud event happens, blob temporarily:
    - loses acceleration, OR
    - maxRun drops, OR
    - input is dampened
- Add a "freeze meter" that spikes and slowly recovers.

Important: keep it short so it's expressive, not frustrating.

---

4) Avoidance steering (body tries to flee even if player doesn't)

- If danger is on the right, blob subtly drifts left (small autopilot bias).
- The player can overpower it, but it feels like the character has its own fear.

---

5) Unsteady jumps (fear affects timing + shape)

- Jump height varies slightly based on panic (small randomness).

- Jump has a brief “crouch” squish before launch (telegraphing nervous prep).
- Mid-air: blob stretches more than usual (like limbs flailing).

---

6) “Startle” impulses

- Environmental triggers (shadow passing, platform creak, falling particle) cause a quick involuntary hop or recoil.
- Works great if it’s rare + predictable enough.

---

7) Breathing becomes visible in physics

- Right now “breathing” is mostly visual (Perlin edge).
- Make fear show up as:
    - slight size pulse affecting collision box (tiny),
    - or micro-vertical bob while grounded,
    - or a subtle “heartbeat” camera shake.

---

8) Stamina / panic sprint

- Holding run too long increases panic.
- At high panic: acceleration increases (desperate sprint) but control decreases (slippery).
- Release run: blob “collapses” into heavier friction and slow movement.

Environment ideas that *cause* fear

1) Darkness / spotlight

- Only a circle around the blob is lit; the rest is dim.
- Threats hinted outside the light (moving silhouettes, eyes, shapes).

This alone makes the same movement feel scarier.

---

2) Unreliable platforms (but fair)

- Some platforms:
    - slightly wobble (visual sway),
    - creak (sound if you add later),
    - or *crumble after you stand for 1 second* (with warning cracks).
- Fear comes from “I can’t stay here.”

3) “Pressure zone” that ramps panic

- Certain areas raise panicLevel (fog, red tint, whispers).
- Panic affects movement (tremble, freeze, overaccel).
- Leaving the zone lets panic decay.

4) Pursuer / looming presence

- A slow, simple enemy that doesn’t kill instantly — it just **forces motion**.
- Even a big shadow sliding across the background that “tracks” you can do it.

5) Sudden environmental events

- Random (but spaced out) events:
  - lights flicker (screen tint pulses),
  - dust falls,
  - a platform drops slightly then resets,
  - distant “thump” shown as a screen shake.

Trigger a brief startle impulse + wobble spike.

6) Claustrophobic layout

- Narrow tunnels, low ceilings, tight landings.
- Fear vibe: you’re squeezed, less room to correct mistakes.
- This pairs well with “skittish acceleration.”

7) The floor isn’t safe

- Spikes aren’t required; you can do “ink” or “void” at floor level.
- If blob touches it: panic spikes + respawn a step back.

Even a non-lethal hazard makes the player tense.

Visual language upgrades (cheap, big impact)

- **Color shift with panic:** blue → desaturated → pale → slightly green/gray.
- **Wobble changes meaning:** calm = slow smooth; fear = faster, sharper, uneven.
- **Squash/stretch:** landings heavier, jumps more “flaily.”

- **Screen feedback:** vignette darkening, subtle shake on scares.

Three strong combos (ready to try in code)

Pick one “package” to implement first.

Package A: “Anxious platformer”

- Tremble idle + faster breathing
- Skittish acceleration
- Darkness spotlight + faint background silhouettes

Package B: “Freeze + flee”

- Panic meter tied to a pursuer distance
- Freeze response when panic spikes
- Platforms that crumble with cracks

Package C: “Unstable world”

- Wobbly platforms + occasional drops
- Startle impulses on events
- Claustrophobic corridors + heavy landing squash

What I’d do first (fastest win)

1. Add a `panic` value (0–1) that increases near a “threat zone” or pursuer.
2. Map panic into:

- wobble amount + wobble speed,
- slight jitter when grounded,
- maxRun / accel changes (small but noticeable),
- optional vignette overlay.

If you tell me which direction you like (A/B/C or a mix), I’ll propose a tight set of code edits (new variables + where to inject them) so we can start testing without rewriting everything.

Side Quests.pdf
PDF
i want to do package A. I also have this document to show you that explains the format i need to work with. Im not great with this coding and github/vscode work so i will need your help with all of it
Thought for 17s

Yep — Package A is a perfect fit for **Week 2** ("redesign the blob's movement and environment to express a specific emotion").

Side Quests

Below is (1) exactly what your submission needs to include (based on your doc), and (2) a clean "Package A" code plan with copy-paste snippets we can drop into your existing sketch.

---

1) The format you need for GBDA 302 Side Quests

What you submit each week

- You submit **two links on LEARN**: your **GitHub repo URL** + **GitHub Pages playable link**.

  Side Quests

- Your repo should follow the naming convention like `YOURUSERNAME_sidequest_W2`.

  Side Quests

Required project structure (must be in the repo)

Your p5 project needs these files/folders at minimum: `index.html`, `sketch.js`, `jsconfig.json`, `style.css`, `libraries/`, and `assets/` (with subfolders like `images/`, `sounds/` if you use them).

Side Quests

README + Process PDF requirements

- README must follow the template (Title, Group Members, Description, Setup/Instructions, Iteration Notes, Assets, References).

  Side Quests

- You must include a short PDF in the repo root named **`Process_and_Decision_Documentation.pdf`** and keep it lightweight + include GenAI transcript if you used GenAI.

  Side Quests

If you want, paste your current repo link for Week 2 (or upload your current folder zip), and I'll tell you exactly what's missing and what to rename.

---

2) Package A implementation (fear): movement + environment

The core vibe

Fear = jittery body + skittish control + darkness spotlight + subtle “presence” silhouettes.

We’ll add one new variable: **panic (0→1)**. Panic rises when:

- you’re in darkness (far from “safe lights”), and/or
- a shadow passes nearby.

Then panic drives:

- faster breathing + higher wobble
- tiny trembling when grounded
- slightly “overreacty” acceleration (jerk), plus a tiny bit of control loss

---

3) Minimal code edits (copy/paste friendly)

A) Add these globals at the top

Paste near your other globals:

```
// --- FEAR SYSTEM (Package A) ---
let panic = 0;              // 0 calm → 1 terrified
let safeLights = [];        // positions of "safe" light pools
let shadows = [];           // background silhouettes
```

B) In setup(), create “safe lights” + shadows

Add inside `setup()` after platforms:

```
safeLights = [
  { x: 80, y: floorY3 - 40, r: 120 },
  { x: 320, y: floorY3 - 140, r: 110 },
  { x: 560, y: floorY3 - 80, r: 120 },
];

// A few drifting shadow blobs in the darkness
for (let i = 0; i < 8; i++) {
  shadows.push({
    x: random(width),
    y: random(height),
    r: random(30, 90),
    vx: random(0.2, 0.8) * random([-1, 1]),
    phase: random(1000),
  });
}
```

C) In draw(), compute panic (before movement is applied)

Near the top of `draw()` after `background(240);` add:

```
// --- FEAR: compute how "safe" we are (distance to nearest light) ---
let nearest = 99999;
for (const L of safeLights) {
  const d = dist(blob3.x, blob3.y, L.x, L.y) - L.r;
  nearest = min(nearest, d);
}
// nearest < 0 means you're inside a safe light pool
let darkness = constrain(map(nearest, -60, 220, 0, 1), 0, 1);

// Panic rises in darkness, calms in light
panic = lerp(panic, darkness, 0.04);
```

D) Make movement “fearful” (tremble + skittish accel)

Replace your acceleration line:

```
blob3.vx += blob3.accel * move;
```

with:

```
// Skittish acceleration: first touch feels a bit jerky when panicked
let skittishBoost = 1 + 0.6 * panic;
blob3.vx += blob3.accel * move * skittishBoost;

// Tremble when grounded + scared (tiny involuntary shake)
if (blob3.onGround && panic > 0.15) {
  blob3.vx += (noise(frameCount * 0.2) - 0.5) * 0.35 * panic;
}
```

Then tweak friction and max speed *based on panic* (right after you apply friction is a good spot). Add:

```
// Fear makes control slightly worse: slippery + a bit slower top speed
let fearFriction = lerp(1.0, 0.94, panic); // lower = more slide
blob3.vx *= (blob3.onGround ? blob3.frictionGround : blob3.frictionAir) *
fearFriction;

let fearMaxRun = lerp(blob3.maxRun, blob3.maxRun * 0.82, panic);
blob3.vx = constrain(blob3.vx, -fearMaxRun, fearMaxRun);
```

(And remove your old `blob3.vx *= ...` and `constrain` lines so you don’t double-apply.)

E) Make the blob “breathe fast” visually

Right before `drawBlobCircle(blob3);` add:

```
// Panic-driven animation: faster breathing + more wobble
blob3.tSpeed = lerp(0.01, 0.05, panic);
```

```
blob3.wobble = lerp(7, 14, panic);
blob3.wobbleFreq = lerp(0.9, 1.6, panic);
```

F) Draw the fearful environment: shadows + spotlight

At the end of `draw()` after you draw the blob and HUD, add this:

```
drawShadows();
drawSpotlight();
```

Then add these helper functions at the bottom:

```
function drawShadows() {
  // subtle drifting silhouettes in the dark
  push();
  noStroke();
  for (const s of shadows) {
    s.x += s.vx;
    if (s.x < -100) s.x = width + 100;
    if (s.x > width + 100) s.x = -100;

    // shadow "breath"
    let rr = s.r + sin(frameCount * 0.02 + s.phase) * 8;

    // darker when panic is high
    fill(0, 30 + 120 * panic);
    ellipse(s.x, s.y, rr * 1.2, rr);
  }
  pop();
}

function drawSpotlight() {
  // dark overlay with a circular "safe" view around the blob
  push();
  fill(0, 170 * (0.35 + 0.65 * panic)); // darker as fear rises
  rect(0, 0, width, height);

  // Cut a hole out of the darkness
  erase();
  let r = lerp(160, 90, panic); // spotlight shrinks when scared
  ellipse(blob3.x, blob3.y, r * 2, r * 2);
  noErase();

  // Optional: slight vignette edge
  fill(0, 60 * panic);
  rect(0, 0, width, height);

  pop();
}
```

That's the "Package A" look in one go: platformer stays playable, but fear is *everywhere*.

---

4) GitHub / VS Code workflow (simple version)

Your doc says you'll code in VS Code and submit via GitHub + Pages.

Side Quests

Side Quests

Here's the straightforward loop:

1. **Create a repo** named `j2metivi_sidequest_W2` (or whatever your username is + week).

   Side Quests

2. Put your p5 files in the repo with the required structure.

   Side Quests

3. In GitHub repo settings → **Pages** → deploy from `main` branch `/ (root)`
4. Test the Pages link in **Chrome** before submitting.

   Side Quests

5. Paste both links into LEARN.

   Side Quests

Next step (so I can guide you cleanly)

To help you "with all of it" without guessing your setup, do one of these:

- Paste your Week 2 repo link (even if it's messy), **or**
- Upload your current project folder (zip), **or**
- Tell me if you're starting from a p5 template / starter repo.

Then I'll:

- tell you exactly what files/folders you're missing (based on the required structure),
- help you set up GitHub Pages,
- and help you write a README that matches the required template.

this is an example of the files that should be in my repo. They are for the example of the blob i gave you. Analyze them and then help me step by step to get this sidequest done
Thought for 4 seconds

Got it. I looked at the example repo files you uploaded — they're a clean "p5.js project template" that matches the structure your Side Quest expects.

What each example file is doing (so you don't fight it)

`index.html`

This is the entry point GitHub Pages loads. It links your CSS and loads p5 from the **libraries/** folder, then runs `sketch.js`.

index

Key detail: it expects these exact paths:

- `libraries/p5.min.js`
- `libraries/p5.sound.min.js`
- `sketch.js`
- `style.css`

So your repo must have a **libraries** folder with those files.

index

`style.css`

Just removes margins so the canvas sits cleanly.

style

`jsconfig.json`

This is for VS Code intellisense / syntax help. It's optional for "running," but good to keep.

jsconfig

`README.md`

This example README shows the format (Title, Authors, Description, etc.). You'll replace it with your own.

README

`sketch.js`

This is the blob + AABB platform collisions example you pasted earlier.

sketch

Step-by-step: getting your Week 2 Side Quest DONE (Package A: Fear)

Step 1 — Create the correct repo structure (exactly like the example)

On your computer, make a folder named:
`j2metivi_sidequest_W2` (or whatever your username is)

Inside it, create:

- `index.html`
- `sketch.js`
- `style.css`
- `jsconfig.json`
- `README.md`
- folder: `libraries/`
- folder: `assets/` (can be empty)

Then put these inside `libraries/`:

- `p5.min.js`
- `p5.sound.min.js`

This matches what `index.html` is expecting.

index

✅ Quick self-check: open `index.html` in VS Code and confirm the script tags look like the example.

index

Step 2 — Put Package A changes into `sketch.js`

You're starting from the example blob code.

sketch

Do these edits **in this order** so you don't break anything:

2A) Add new globals near the top (above `setup()`)

Paste right under `let platforms = [];`:

```
// --- FEAR SYSTEM (Package A) ---
let panic = 0;            // 0 calm → 1 terrified
let safeLights = [];      // safe light pools
let shadows = [];         // drifting silhouettes
```

2B) In `setup()`, add safe lights + shadows

Paste near the end of `setup()` (after you set `blob3.y` is fine):

```
safeLights = [
  { x: 80, y: floorY3 - 40, r: 120 },
  { x: 320, y: floorY3 - 140, r: 110 },
  { x: 560, y: floorY3 - 80, r: 120 },
];

for (let i = 0; i < 8; i++) {
  shadows.push({
    x: random(width),
    y: random(height),
    r: random(30, 90),
    vx: random(0.2, 0.8) * random([-1, 1]),
    phase: random(1000),
  });
}
```

2C) In `draw()`, compute panic near the top

Right after `background(240);` add:

```
let nearest = 99999;
for (const L of safeLights) {
  const d = dist(blob3.x, blob3.y, L.x, L.y) - L.r;
  nearest = min(nearest, d);
}
let darkness = constrain(map(nearest, -60, 220, 0, 1), 0, 1);
panic = lerp(panic, darkness, 0.04);
```

2D) Replace the movement + friction lines (important)

Find this in your file (it's in the example):

sketch

```
blob3.vx += blob3.accel * move;
blob3.vx *= blob3.onGround ? blob3.frictionGround : blob3.frictionAir;
blob3.vx = constrain(blob3.vx, -blob3.maxRun, blob3.maxRun);
```

Replace that whole block with:

```
// Skittish acceleration (fear makes inputs feel jerky)
let skittishBoost = 1 + 0.6 * panic;
blob3.vx += blob3.accel * move * skittishBoost;

// Tremble when grounded + scared
if (blob3.onGround && panic > 0.15) {
  blob3.vx += (noise(frameCount * 0.2) - 0.5) * 0.35 * panic;
}

// Fear makes control slightly worse (more slide) + lower top speed
let fearFriction = lerp(1.0, 0.94, panic);
blob3.vx *= (blob3.onGround ? blob3.frictionGround : blob3.frictionAir) *
fearFriction;

let fearMaxRun = lerp(blob3.maxRun, blob3.maxRun * 0.82, panic);
blob3.vx = constrain(blob3.vx, -fearMaxRun, fearMaxRun);
```

2E) Make the blob “breathe fast” visually

Right before `drawBlobCircle(blob3);`, add:

```
blob3.tSpeed = lerp(0.01, 0.05, panic);
blob3.wobble = lerp(7, 14, panic);
blob3.wobbleFreq = lerp(0.9, 1.6, panic);
```

2F) Add the fear environment overlay (spotlight + shadows)

At the end of `draw()` (after HUD text), add:

```
drawShadows();
drawSpotlight();
```

Then add these functions near the bottom of the file:

```
function drawShadows() {
  push();
  noStroke();
  for (const s of shadows) {
    s.x += s.vx;
    if (s.x < -100) s.x = width + 100;
    if (s.x > width + 100) s.x = -100;

    let rr = s.r + sin(frameCount * 0.02 + s.phase) * 8;
    fill(0, 30 + 120 * panic);
    ellipse(s.x, s.y, rr * 1.2, rr);
  }
  pop();
}

function drawSpotlight() {
  push();
  fill(0, 170 * (0.35 + 0.65 * panic));
  rect(0, 0, width, height);

  erase();
  let r = lerp(160, 90, panic);
  ellipse(blob3.x, blob3.y, r * 2, r * 2);
  noErase();

  fill(0, 60 * panic);
  rect(0, 0, width, height);
  pop();
}
```

✅ After this, your blob should feel “anxious” AND the world should feel “unsafe.”

Step 3 — Update `README.md` (use the example format)

Your README should follow the structure shown in the example.

README

Replace “Authors” with you, and description should say you redesigned movement + environment to express **fear** (Package A).

If you paste your current README draft, I’ll rewrite it in the right format quickly.

Step 4 — Create your Process PDF (required)

You uploaded the Process Document template and it’s the right one.

Process Document

Export it as:
`Process_and_Decision_Documentation.pdf`

Put it in the **root** of the repo (same level as `index.html`).

Step 5 — Upload to GitHub (simple way)

1. On GitHub → create a new repo named `j2metivi_sidequest_W2`
2. Upload your folder contents (or use GitHub Desktop if you prefer)
3. Confirm repo shows:
    - `index.html`, `sketch.js`, `style.css`, `README.md`, `jsconfig.json`
    - `libraries/` with p5 files
    - `assets/`
    - `Process_and_Decision_Documentation.pdf`

Step 6 — Turn on GitHub Pages

Repo → Settings → Pages:

- Source: Deploy from a branch
- Branch: `main`
- Folder: `/ (root)`

Because your `index.html` is in root and it loads scripts from `libraries/`, this setup matches the example.

index

Step 7 — Final checks before submission

Open your GitHub Pages link and confirm:

- The canvas shows up (no white page)
- Blob moves + jumps
- Darkness/spotlight works
- Tremble is noticeable in darkness

Your next move (tell me ONE of these and I'll guide you precisely)

1. Paste your **current Week 2 repo link** (even if incomplete) and I'll tell you exactly what's missing.
2. Or tell me: are you using **VS Code + Git** or **uploading files through GitHub website**?

Either way, we'll finish it cleanly.

this is the readme.me file starter they gave us. ## Project Title Give your project a short, clear title (e.g., "Side Quest Week 1: Interactive Shape Switcher"). --- ## Group Number (only for group work) List your group number. --- ## Description Briefly describe what your project does. Explain what the user will see, how they can interact, and what inspired the design. Example: > This project lets users switch between shapes using keyboard inputs. > It explores how visual feedback can communicate accessibility states. --- ## Interaction Instructions Describe how to use or play your project. - How does the user start? - What controls or keys do they use? - Any important visual or audio feedback to watch for? Example: > Press `S` to switch shapes and `R` to reset the canvas. --- ## Assets List any images, sounds, or other files used in your assets/ folder. Include in-text citations for all assets that are not your own, and connect them to your reference list. Example: > - `assets/sound/beep.mp3` (created in GarageBand) > - `assets/image/background.png` (original artwork) --- ## References Connect your in-text citations to the reference list along with additional sources that informed your design but were not directly cited. Provide all citations using the ACM reference format. ---

Perfect — this is a **good, clean starter README**, and for a **Side Quest** you don't need to overthink it. I'll do two things:

1. **Fill this README out for your specific project (Package A: Fear Blob)**
2. Explain **why each section is written this way**, so you understand it (and can tweak it if needed)

You can copy-paste the README below directly into `README.md`.

✅ Final README.md (ready to submit)

```
## Project Title
Side Quest Week 2: Anxious Blob Platformer

---
## Group Number (only for group work)
N/A – Individual Side Quest

---
## Description
This project is a p5.js platformer featuring a soft, animated blob whose
movement and environment have been redesigned to express fear.
The blob becomes unstable and jittery in dark spaces, with movement that feels
skittish and hesitant rather than confident.
Environmental lighting and drifting background shadows reinforce a sense of
unease without introducing enemies or combat, focusing instead on emotional
expression through motion and atmosphere.
```

```
---
## Interaction Instructions
- Use `A / D` or `← / →` to move left and right
- Press `W`, `↑`, or `Space` to jump (only when grounded)
- Move toward lighted areas to calm the blob
- Darkness increases the blob’s panic, making movement less stable and
visibility more limited

Watch how the blob’s body deformation and movement change based on its
emotional state.

---
## Assets
No external image or sound assets were used in this project.
All visuals are generated procedurally using p5.js.

---
## References
Processing Foundation. *p5.js Reference*. https://p5js.org/reference/
```

🧠 Why this README works (quick explanation)

Project Title

- Includes **Side Quest + week number** → clear for graders
- Names the **emotion** explicitly (“Anxious Blob”) → aligns with assignment

Group Number

- “N/A – Individual Side Quest” is **explicit and safe**
- Prevents confusion during marking

Description

This hits exactly what they want:

- What it is (p5.js platformer)
- What the user experiences
- What the design focus is (fear via movement + environment)
- No unnecessary technical detail

Interaction Instructions

- Bullet points = readable
- Mentions **emotional feedback**, not just controls
- Shows you’re thinking like a designer, not just a coder

Assets

- Saying *“No external assets”* is totally valid
- Better than leaving it blank

References

- p5.js reference is enough here
- You are **not required** to cite ChatGPT in README (that goes in the Process PDF)

✅ Where this file goes

Place `README.md` in the **root of your repo**, alongside:

```
index.html
sketch.js
style.css
jsconfig.json
libraries/
assets/
Process_and_Decision_Documentation.pdf
```

Next step (tell me which one)

We’re very close to “done-done.”
Pick **one** and I’ll walk you through it:

1. “Check my repo structure”
2. “Help me enable GitHub Pages”
3. “Test my sketch for bugs / fear readability”
4. “Final submission checklist before I upload”

You’re doing really well — this is exactly the level they expect for a Side Quest.

i made my repo folder. Can you give me the full code for my sketch.js

Yep. Here’s a full `sketch.js` you can paste in **as-is**. It’s your original blob platformer, redesigned for **Package A (fear)**:

- Panic system (0–1) based on distance to safe lights
- Skittish acceleration + trembling when scared
- Faster “breathing” + more wobble when scared
- Drifting shadow silhouettes
- Darkness overlay with a shrinking spotlight around the blob

```
// Y-position of the floor (ground level)
```

```
let floorY3;

// Player character (soft, animated blob)
let blob3 = {
  // Position (centre of the blob)
  x: 80,
  y: 0,

  // Visual properties
  r: 26, // Base radius
  points: 48, // Number of points used to draw the blob
  wobble: 7, // Edge deformation amount
  wobbleFreq: 0.9,

  // Time values for breathing animation
  t: 0,
  tSpeed: 0.01,

  // Physics: velocity
  vx: 0, // Horizontal velocity
  vy: 0, // Vertical velocity

  // Movement tuning
  accel: 0.55, // Horizontal acceleration
  maxRun: 4.0, // Maximum horizontal speed
  gravity: 0.65, // Downward force
  jumpV: -11.0, // Initial jump impulse

  // State
  onGround: false, // True when standing on a platform

  // Friction
  frictionAir: 0.995, // Light friction in air
  frictionGround: 0.88, // Stronger friction on ground
};

// List of solid platforms the blob can stand on
// Each platform is an axis-aligned rectangle (AABB)
let platforms = [];

// --- FEAR SYSTEM (Package A) ---
let panic = 0; // 0 calm → 1 terrified
let safeLights = []; // positions of "safe" light pools
let shadows = []; // background silhouettes

function setup() {
  createCanvas(640, 360);

  // Define the floor height
  floorY3 = height - 36;

  noStroke();
  textFont("sans-serif");
  textSize(14);

  // Create platforms (floor + steps)
  platforms = [
    { x: 0, y: floorY3, w: width, h: height - floorY3 }, // floor
    { x: 120, y: floorY3 - 70, w: 120, h: 12 }, // low step
    { x: 300, y: floorY3 - 120, w: 90, h: 12 }, // mid step
    { x: 440, y: floorY3 - 180, w: 130, h: 12 }, // high step
    { x: 520, y: floorY3 - 70, w: 90, h: 12 }, // return ramp
  ];

  // Start the blob resting on the floor
  blob3.y = floorY3 - blob3.r - 1;

  // Safe light pools (calm zones)
  safeLights = [
    { x: 80, y: floorY3 - 40, r: 120 },
    { x: 320, y: floorY3 - 140, r: 110 },
    { x: 560, y: floorY3 - 80, r: 120 },
  ];

  // Drifting background shadows (unease in the dark)
  for (let i = 0; i < 8; i++) {
    shadows.push({
      x: random(width),
      y: random(height),
      r: random(30, 90),
      vx: random(0.2, 0.8) * random([-1, 1]),
      phase: random(1000),
    });
  }
}

function draw() {
  background(240);

  // --- FEAR: compute how "safe" we are (distance to nearest light) ---
  let nearest = 99999;
  for (const L of safeLights) {
    const d = dist(blob3.x, blob3.y, L.x, L.y) - L.r;
    nearest = min(nearest, d);
  }
  // nearest < 0 means inside a safe light pool
  let darkness = constrain(map(nearest, -60, 220, 0, 1), 0, 1);

  // Panic rises in darkness, calms in light
  panic = lerp(panic, darkness, 0.04);

  // --- Draw all platforms (base layer) ---
  fill(200);
  for (const p of platforms) {
    rect(p.x, p.y, p.w, p.h);
  }

  // --- Input: left/right movement ---
  let move = 0;
  if (keyIsDown(65) || keyIsDown(LEFT_ARROW)) move -= 1; // A or ←
  if (keyIsDown(68) || keyIsDown(RIGHT_ARROW)) move += 1; // D or →
  // Skittish acceleration (fear makes inputs feel jerky)
  let skittishBoost = 1 + 0.6 * panic;
  blob3.vx += blob3.accel * move * skittishBoost;
  // Tremble when grounded + scared (tiny involuntary shake)
  if (blob3.onGround && panic > 0.15) {
    blob3.vx += (noise(frameCount * 0.2) - 0.5) * 0.35 * panic;
  }

  // --- Apply friction and clamp speed (panic affects control) ---
  let fearFriction = lerp(1.0, 0.94, panic); // lower = more slide
  blob3.vx *= (blob3.onGround ? blob3.frictionGround : blob3.frictionAir) *
fearFriction;

  let fearMaxRun = lerp(blob3.maxRun, blob3.maxRun * 0.82, panic);
  blob3.vx = constrain(blob3.vx, -fearMaxRun, fearMaxRun);

  // --- Apply gravity ---
  blob3.vy += blob3.gravity;

  // --- Collision representation ---
  // We collide using a rectangle (AABB),
  // even though the blob is drawn as a circle
  let box = {
    x: blob3.x - blob3.r,
    y: blob3.y - blob3.r,
    w: blob3.r * 2,
    h: blob3.r * 2,
  };

  // --- STEP 1: Move horizontally, then resolve X collisions ---
  box.x += blob3.vx;
  for (const s of platforms) {
    if (overlap(box, s)) {
      if (blob3.vx > 0) {
        // Moving right → hit the left side of a platform
        box.x = s.x - box.w;
      } else if (blob3.vx < 0) {
        // Moving left → hit the right side of a platform
        box.x = s.x + s.w;
      }
      blob3.vx = 0;
    }
  }

  // --- STEP 2: Move vertically, then resolve Y collisions ---
  box.y += blob3.vy;
  blob3.onGround = false;

  for (const s of platforms) {
    if (overlap(box, s)) {
      if (blob3.vy > 0) {
        // Falling → land on top of a platform
        box.y = s.y - box.h;
        blob3.vy = 0;
        blob3.onGround = true;
      } else if (blob3.vy < 0) {
        // Rising → hit the underside of a platform
        box.y = s.y + s.h;
        blob3.vy = 0;
      }
    }
  }

  // --- Convert collision box back to blob centre ---
  blob3.x = box.x + box.w / 2;
  blob3.y = box.y + box.h / 2;

  // Keep blob inside the canvas horizontally
  blob3.x = constrain(blob3.x, blob3.r, width - blob3.r);

  // --- Panic-driven animation: faster breathing + more wobble ---
  blob3.tSpeed = lerp(0.01, 0.05, panic);
  blob3.wobble = lerp(7, 14, panic);
  blob3.wobbleFreq = lerp(0.9, 1.6, panic);

  // --- Draw the animated blob ---
  blob3.t += blob3.tSpeed;
  drawBlobCircle(blob3);

  // --- HUD ---
  fill(0);
  text("Move: A/D or ←/→  •  Jump: Space/W/↑", 10, 18);
  text("Stay in light to calm down. Darkness increases panic.", 10, 36);

  // --- Fear environment overlay (draw last) ---
  drawShadows();
  drawSpotlight();
}

// Axis-Aligned Bounding Box (AABB) overlap test
// Returns true if rectangles a and b intersect
function overlap(a, b) {
  return (
    a.x < b.x + b.w &&
    a.x + a.w > b.x &&
    a.y < b.y + b.h &&
    a.y + a.h > b.y
  );
}

// Draws the blob using Perlin noise for a soft, breathing effect
function drawBlobCircle(b) {
  fill(20, 120, 255);
  beginShape();

  for (let i = 0; i < b.points; i++) {
    const a = (i / b.points) * TAU;

    // Noise-based radius offset
    const n = noise(
      cos(a) * b.wobbleFreq + 100,
      sin(a) * b.wobbleFreq + 100,
      b.t
    );

    const r = b.r + map(n, 0, 1, -b.wobble, b.wobble);
    vertex(b.x + cos(a) * r, b.y + sin(a) * r);
  }

  endShape(CLOSE);
}

// Jump input (only allowed when grounded)
function keyPressed() {
  if (
    (key === " " || key === "W" || key === "w" || keyCode === UP_ARROW) &&
    blob3.onGround
  ) {
    blob3.vy = blob3.jumpV;
    blob3.onGround = false;
  }
}

// --- Fear ambience: drifting silhouettes ---
function drawShadows() {
  push();
  noStroke();
  for (const s of shadows) {
    s.x += s.vx;
    if (s.x < -100) s.x = width + 100;
    if (s.x > width + 100) s.x = -100;

    // shadow "breath"
    let rr = s.r + sin(frameCount * 0.02 + s.phase) * 8;

    // darker when panic is high
    fill(0, 30 + 120 * panic);
    ellipse(s.x, s.y, rr * 1.2, rr);
  }
  pop();
}

// --- Fear ambience: darkness overlay + spotlight around blob ---
function drawSpotlight() {
  push();

  // Dark overlay
  fill(0, 170 * (0.35 + 0.65 * panic)); // darker as fear rises
  rect(0, 0, width, height);

  // Cut a hole out of the darkness
  erase();
  let r = lerp(160, 90, panic); // spotlight shrinks when scared
  ellipse(blob3.x, blob3.y, r * 2, r * 2);
  noErase();

  // Extra vignette feel
  fill(0, 60 * panic);
  rect(0, 0, width, height);

  pop();
}
```

After you paste it

Quick test:

1. Run your project locally (or open `index.html` in a live server).
2. Move into darker areas → spotlight shrinks + blob trembles and breathes faster.

If anything goes wrong, tell me what you see (blank page? console error? spotlight not working?) and I'll fix it fast.

the blob is huge and there isnt really even dark or light areas, its all just the same. The hitbox for the blob is also smaller than the blob so the majority of it can go through the platforms but it only recognizes when the center of the blob hits them
Thought for 15s

Totally fair feedback — both issues come from **(1) visuals not actually showing "light pools"** and **(2) collisions using `blob3.r` while the *drawn* blob can be larger because of wobble**.

What's happening

- **"No light/dark areas"**: In the version I gave you, the whole screen gets a darkness overlay + a blob spotlight hole, but we never actually *drawed visible light pools* in the world. So it reads like "same everywhere."
- **Hitbox smaller than blob**: Your collision box is based on `blob3.r`, but the blob is drawn with `r + wobble` (and wobble increases with panic). So visually it can overlap platforms even though the collision thinks it hasn't touched yet.

Fix (simple + solid): make light pools visible + make hitbox match blob size

Below is a **full replacement `sketch.js`** that:

- draws **real visible light pools** (soft glowy circles)
- makes the collision radius = `blob3.r + blob3.wobble` so the hitbox matches what you see
- tones down wobble so it doesn't feel "huge"
- keeps the fear vibe: trembling + skittish acceleration + faster breathing in darkness

Paste this whole file in as-is:

```
// Y-position of the floor (ground level)
let floorY3;

// Player character (soft, animated blob)
let blob3 = {
  x: 80,
  y: 0,
  r: 26,
  points: 48,
  wobble: 6,
  wobbleFreq: 0.9,

  t: 0,
  tSpeed: 0.012,

  vx: 0,
  vy: 0,

  accel: 0.55,
  maxRun: 4.0,
  gravity: 0.65,
  jumpV: -11.0,

  onGround: false,

  frictionAir: 0.995,
  frictionGround: 0.88,
};

let platforms = [];

// --- FEAR SYSTEM (Package A) ---
let panic = 0; // 0 calm → 1 terrified
let safeLights = [];
let shadows = [];

function setup() {
  createCanvas(640, 360);
  floorY3 = height - 36;

  noStroke();
  textFont("sans-serif");
  textSize(14);

  platforms = [
    { x: 0, y: floorY3, w: width, h: height - floorY3 }, // floor
    { x: 120, y: floorY3 - 70, w: 120, h: 12 },
    { x: 300, y: floorY3 - 120, w: 90, h: 12 },
    { x: 440, y: floorY3 - 180, w: 130, h: 12 },
    { x: 520, y: floorY3 - 70, w: 90, h: 12 },
  ];

  blob3.y = floorY3 - blob3.r - 1;

  // Visible safe light pools (these are what create “light areas”)
  safeLights = [
    { x: 80, y: floorY3 - 40, r: 130 },
    { x: 320, y: floorY3 - 140, r: 120 },
    { x: 560, y: floorY3 - 80, r: 130 },
  ];

  for (let i = 0; i < 8; i++) {
    shadows.push({
      x: random(width),
      y: random(height),
      r: random(40, 120),
      vx: random(0.15, 0.55) * random([-1, 1]),
      phase: random(1000),
    });
  }
}

function draw() {
  // Base “dark world”
  background(18);

  // Shadows first (behind everything)
  drawShadows();

  // Draw visible light pools (THIS makes light/dark readable)
  drawLightPools();

  // Panic = how far outside light pools the blob is
  let nearest = 99999;
  for (const L of safeLights) {
    const d = dist(blob3.x, blob3.y, L.x, L.y) - L.r;
    nearest = min(nearest, d);
  }
  // nearest < 0 means you're in light
  let darkness = constrain(map(nearest, -60, 260, 0, 1), 0, 1);
  panic = lerp(panic, darkness, 0.05);

  // Panic-driven animation tuning (keep wobble reasonable)
  blob3.tSpeed = lerp(0.012, 0.05, panic);
  blob3.wobble = lerp(5, 9, panic);
  blob3.wobbleFreq = lerp(0.9, 1.5, panic);

  // --- Draw platforms (slightly dim so lights stand out) ---
  fill(70);
  for (const p of platforms) rect(p.x, p.y, p.w, p.h);

  // --- Input ---
  let move = 0;
  if (keyIsDown(65) || keyIsDown(LEFT_ARROW)) move -= 1;
  if (keyIsDown(68) || keyIsDown(RIGHT_ARROW)) move += 1;

  // Skittish acceleration: fear makes the first push a bit sharper
  let skittishBoost = 1 + 0.5 * panic;
  blob3.vx += blob3.accel * move * skittishBoost;

  // Tremble when grounded + scared (small involuntary shake)
  if (blob3.onGround && panic > 0.15) {
    blob3.vx += (noise(frameCount * 0.25) - 0.5) * 0.28 * panic;
  }

  // Fear slightly reduces control (more slide) + slightly reduces top speed
  let fearFriction = lerp(1.0, 0.95, panic);
  blob3.vx *= (blob3.onGround ? blob3.frictionGround : blob3.frictionAir) *
fearFriction;
```

```
let fearMaxRun = lerp(blob3.maxRun, blob3.maxRun * 0.85, panic);
blob3.vx = constrain(blob3.vx, -fearMaxRun, fearMaxRun);

// Gravity
blob3.vy += blob3.gravity;

// --- IMPORTANT FIX: collision radius matches visual blob size ---
// The blob is drawn up to (r + wobble), so collide using that.
let collisionR = blob3.r + blob3.wobble;

let box = {
  x: blob3.x - collisionR,
  y: blob3.y - collisionR,
  w: collisionR * 2,
  h: collisionR * 2,
};

// Move X + resolve
box.x += blob3.vx;
for (const s of platforms) {
  if (overlap(box, s)) {
    if (blob3.vx > 0) box.x = s.x - box.w;
    else if (blob3.vx < 0) box.x = s.x + s.w;
    blob3.vx = 0;
  }
}

// Move Y + resolve
box.y += blob3.vy;
blob3.onGround = false;

for (const s of platforms) {
  if (overlap(box, s)) {
    if (blob3.vy > 0) {
      box.y = s.y - box.h;
      blob3.vy = 0;
      blob3.onGround = true;
    } else if (blob3.vy < 0) {
      box.y = s.y + s.h;
      blob3.vy = 0;
    }
  }
}

// Convert back to center position
blob3.x = box.x + box.w / 2;
blob3.y = box.y + box.h / 2;

// Keep inside canvas horizontally using collision radius
blob3.x = constrain(blob3.x, collisionR, width - collisionR);

// Draw blob
blob3.t += blob3.tSpeed;
drawBlobCircle(blob3, panic);

// Optional vignette (adds tension without killing the light pools)
drawVignette();
```

```
  // HUD
  fill(220);
  text("Move: A/D or ←/→  •  Jump: Space/W/↑", 10, 18);
  text("Stay in light to calm down. Darkness increases panic.", 10, 36);
}

// AABB overlap
function overlap(a, b) {
  return (
    a.x < b.x + b.w &&
    a.x + a.w > b.x &&
    a.y < b.y + b.h &&
    a.y + a.h > b.y
  );
}

// Blob drawing (color shifts slightly with panic)
function drawBlobCircle(b, panicAmt) {
  // More fear = less saturated / paler
  let baseR = lerp(30, 200, 1 - panicAmt);
  let baseG = lerp(120, 210, 1 - panicAmt);
  let baseB = lerp(255, 230, 1 - panicAmt);
  fill(baseR, baseG, baseB);

  beginShape();
  for (let i = 0; i < b.points; i++) {
    const a = (i / b.points) * TAU;
    const n = noise(cos(a) * b.wobbleFreq + 100, sin(a) * b.wobbleFreq + 100,
b.t);
    const r = b.r + map(n, 0, 1, -b.wobble, b.wobble);
    vertex(b.x + cos(a) * r, b.y + sin(a) * r);
  }
  endShape(CLOSE);
}

function keyPressed() {
  if ((key === " " || key === "W" || key === "w" || keyCode === UP_ARROW) &&
blob3.onGround) {
    blob3.vy = blob3.jumpV;
    blob3.onGround = false;
  }
}

// Drift silhouettes (subtle fear texture)
function drawShadows() {
  push();
  noStroke();
  for (const s of shadows) {
    s.x += s.vx;
    if (s.x < -150) s.x = width + 150;
    if (s.x > width + 150) s.x = -150;

    let rr = s.r + sin(frameCount * 0.02 + s.phase) * 10;
    fill(0, 50);
    ellipse(s.x, s.y, rr * 1.2, rr);
  }
  pop();
}

// Visible light pools (so you clearly see safe zones)
function drawLightPools() {
  push();
  noStroke();
  for (const L of safeLights) {
    // Draw a simple radial glow using layered circles
    for (let i = 12; i >= 1; i--) {
      let t = i / 12;
      let rr = L.r * (1.0 + (1 - t) * 0.25);
      let alpha = 18 * t;
      fill(255, 240, 200, alpha);
      ellipse(L.x, L.y, rr * 2, rr * 2);
    }

    // bright core
    fill(255, 245, 210, 55);
    ellipse(L.x, L.y, L.r * 0.9, L.r * 0.9);
  }
  pop();
}

// Vignette that strengthens with panic (doesn't erase lights)
function drawVignette() {
  push();
  noStroke();
  fill(0, 90 * panic);
  rect(0, 0, width, height);
  pop();
}
```

Quick “sanity checks” (so you know it’s fixed)

1. **Blob size:** should look normal again (no huge wobble).
2. **Light/dark:** you should clearly see bright pools at the three light spots.
3. **Hitbox:** the blob should no longer “sink into” platforms—collision uses `r + wobble`.

If you want it EVEN clearer (optional tweaks)

- Make lights stronger: in `drawLightPools()`, increase alpha numbers (`18`, `55`).
- Make darkness scarier: change `background(18)` to `background(10)`.
- Make panic more intense: change `panic = lerp(panic, darkness, 0.05);` to `0.08`.

If you run this and something still feels off, tell me *which one*:

- “Lights are too subtle”

- “Blob still clips”
- “Movement doesn’t feel scared enough”
- “It’s too hard to play”

…and I’ll tune the exact values with you.

everything is looking real good right now. The issue i have found is that the hitbox for the blob is now bigger than the blob itself so it cant really go anywhere because of that

Nice — that means we overshot the collision radius.

Right now the collision box is using:

- `collisionR = blob3.r + blob3.wobble`

…but **wobble is “edge noise”, not the blob’s true silhouette everywhere**, and your glow/shape makes the blob *look* smaller than its maximum noisy radius. So the hitbox ends up feeling bigger than what your eye sees.

Fix: use a “visual radius” that’s slightly *less* than r+wobble

Instead of colliding at the maximum possible blob edge, we’ll collide at:

`collisionR = blob3.r + blob3.wobble * 0.35`

That keeps the hitbox close to the average outline, while still preventing most clipping.

✅ Make this one edit

In your `draw()` where you currently have:

```
let collisionR = blob3.r + blob3.wobble;
```

replace with:

```
// Hitbox tuned to match perceived size (not max wobble)
let collisionR = blob3.r + blob3.wobble * 0.35;
```

That’s it.

If you want it *perfect*, add a debug toggle (optional)

This draws the collision box so you can visually match it:

1. Add this global at the top:

```
let showHitbox = false;
```

2. In `keyPressed()`, add:

```
if (key === "h" || key === "H") showHitbox = !showHitbox;
```

3. After you compute `box` and update blob position, add:

```
if (showHitbox) {
  push();
  noFill();
  stroke(255, 0, 0);
  rect(box.x, box.y, box.w, box.h);
  pop();
  noStroke();
}
```

Now press **H** to see exactly what the collision thinks.

Quick tuning guide

If it's still too big: change `0.35` → `0.25`
If it clips again: change `0.35` → `0.45`

If you want, paste the few lines around where you compute `collisionR` and `box`, and I'll tell you exactly where to drop the debug block so it doesn't mess with your drawing order.

im confused with step 3

Totally fair — let's slow this *way* down and make step 3 idiot-proof 😄
You're actually very close.

I'll explain **exactly where** things go, using **landmarks from your file**, not abstractions.

What step 3 is trying to do (plain English)

Step 3 **just draws a red rectangle** so you can *see* the hitbox.
It does **NOT** change physics or behavior.

You can even skip it — it's only for understanding/debugging.

Step-by-step, zero guessing

✅ STEP 3A — Find THIS section in your `draw()` function

Scroll until you see **this exact comment** (it's already in your file):

```
// Convert back to center position
blob3.x = box.x + box.w / 2;
blob3.y = box.y + box.h / 2;
```

This is the **correct anchor point**.
The hitbox has already been calculated at this point.

✅ STEP 3B — Paste this IMMEDIATELY after it

Copy-paste **this whole block** directly underneath those two lines:

```
// DEBUG: draw hitbox (press H to toggle)
if (showHitbox) {
  push();
  noFill();
  stroke(255, 0, 0);
  rect(box.x, box.y, box.w, box.h);
  pop();
  noStroke();
}
```

So it should look like this when you're done:

```
// Convert back to center position
blob3.x = box.x + box.w / 2;
blob3.y = box.y + box.h / 2;

// DEBUG: draw hitbox (press H to toggle)
if (showHitbox) {
  push();
  noFill();
  stroke(255, 0, 0);
  rect(box.x, box.y, box.w, box.h);
  pop();
  noStroke();
}
```

✅ You're done with Step 3B.

✅ STEP 3C — Add the toggle (ONE line)

At the **very top of the file**, with your other globals, add:

```
let showHitbox = false;
```

Now scroll down to **this function** (already in your file):

```
function keyPressed() {
  if ((key === " " || key === "W" || key === "w" || keyCode === UP_ARROW) &&
blob3.onGround) {
    blob3.vy = blob3.jumpV;
    blob3.onGround = false;
  }
}
```

Add **this ONE line inside the function** (anywhere):

```
if (key === "h" || key === "H") showHitbox = !showHitbox;
```

Final version:

```
function keyPressed() {
  if ((key === " " || key === "W" || key === "w" || keyCode === UP_ARROW) &&
blob3.onGround) {
    blob3.vy = blob3.jumpV;
    blob3.onGround = false;
  }

  if (key === "h" || key === "H") showHitbox = !showHitbox;
}
```

How to use it

1. Run the sketch
2. Press **H**
3. You'll see a **red box**
4. That red box is the collision hitbox
5. Adjust this line to tune it:

```
let collisionR = blob3.r + blob3.wobble * 0.35;
```

Tuning cheat sheet

- Too big → `0.35 → 0.25`
- Still blocking movement → `0.25 → 0.2`
- Clipping through platforms → raise it slightly

Reassurance (important)

You didn't misunderstand anything fundamental — this confusion is **normal** because:

- collision math is invisible

- wobble is visual noise, not geometry
- p5 doesn't show hitboxes by default

You're doing exactly what designers *should* do: test, notice mismatch, fix.

If you want, next we can:

- lock the collision radius permanently and remove debug
- clean up comments for submission
- do a final "fear readability" polish pass

Just tell me 👍

theres no issue with my hitbox. There is another issue. When the blob goes a certain distance over to the right, it glitches and shoots back to the left corner where it started. Im not too sure why its doing this. It does it when i try to land on the leftmost platform and also when i go underneath it
Thought for 24s

Ahhh yep — that specific "shoot back left" behavior is almost always this:

What's actually happening

Your **floor is a giant rectangle** (`{ x:0, y:floorY3, w:width, h:height-floorY3 }`).
If the blob's collision box *even slightly* overlaps that floor rectangle during the **horizontal collision step**, your X-resolution code treats the floor like a wall and does:

- moving right → `box.x = s.x - box.w` → becomes negative
- then your `constrain()` pushes the blob to the left side

So it *looks like the blob teleports back left*.

This can happen:

- when landing near platform edges (tiny overlap)
- when going under a platform and getting nudged downward for a frame
- when the wobble / panic changes the radius slightly and creates a 1-frame overlap with the floor

Fix (best + simplest): Skip the floor in the X-collision loop

The floor should block you **vertically**, not sideways.

Step 1 — Mark the floor in your platforms array

In `setup()`, change the floor platform to include a flag:

```
platforms = [
  { x: 0, y: floorY3, w: width, h: height - floorY3, isFloor: true }, // floor
  { x: 120, y: floorY3 - 70, w: 120, h: 12 },
  { x: 300, y: floorY3 - 120, w: 90, h: 12 },
  { x: 440, y: floorY3 - 180, w: 130, h: 12 },
  { x: 520, y: floorY3 - 70, w: 90, h: 12 },
];
```

Step 2 — Skip the floor during horizontal collision resolution

Find this block in `draw()`:

```
// Move X + resolve
box.x += blob3.vx;
for (const s of platforms) {
  if (overlap(box, s)) {
    if (blob3.vx > 0) box.x = s.x - box.w;
    else if (blob3.vx < 0) box.x = s.x + s.w;
    blob3.vx = 0;
  }
}
```

Replace it with:

```
// Move X + resolve (skip floor so it doesn't act like a side wall)
box.x += blob3.vx;
for (const s of platforms) {
  if (s.isFloor) continue;
  if (overlap(box, s)) {
    if (blob3.vx > 0) box.x = s.x - box.w;
    else if (blob3.vx < 0) box.x = s.x + s.w;
    blob3.vx = 0;
  }
}
```

That’s it. This will stop the “teleport left” glitch.

Optional extra safety (if it still happens rarely)

Right after your **vertical collision loop**, add this line to guarantee you never remain inside the floor:

```
box.y = min(box.y, floorY3 - box.h);
```

(But in most cases, skipping the floor in X is enough.)

If you want, paste your current “Move X + resolve” and your `platforms = [...]` block, and I’ll point out exactly where to drop each change in *your* file.

where exactly am i adding the optional extra safety?

Good question — this one is **very specific**, so I’ll anchor it to exact lines in *your* file.

You only add the **optional extra safety line** in **one place**, and it’s **inside `draw()`**, **after** vertical collisions are resolved, but **before** you convert the box back to the blob’s center.

Step-by-step (no guessing)

1️⃣ Find this block in `draw()`

Look for this exact comment (it’s already in your file):

```
// Move Y + resolve
box.y += blob3.vy;
blob3.onGround = false;

for (const s of platforms) {
  if (overlap(box, s)) {
    if (blob3.vy > 0) {
      box.y = s.y - box.h;
      blob3.vy = 0;
      blob3.onGround = true;
    } else if (blob3.vy < 0) {
      box.y = s.y + s.h;
      blob3.vy = 0;
    }
  }
}
```

2️⃣ ADD THIS LINE immediately after that loop

Paste **this single line** right underneath the `for` loop:

```
// Safety clamp: never allow box to remain inside the floor
box.y = min(box.y, floorY3 - box.h);
```

So it should now look like this:

```
for (const s of platforms) {
  if (overlap(box, s)) {
    if (blob3.vy > 0) {
      box.y = s.y - box.h;
      blob3.vy = 0;
      blob3.onGround = true;
    } else if (blob3.vy < 0) {
      box.y = s.y + s.h;
      blob3.vy = 0;
    }
  }
}

// Safety clamp: never allow box to remain inside the floor
box.y = min(box.y, floorY3 - box.h);
```

3️⃣Do NOT put it anywhere else

❌ Not before the Y loop
❌ Not inside the loop
❌ Not after you convert back to `blob3.x / y`

It **must** be:

after vertical collision resolution
before this section ↓

```
// Convert back to center position
blob3.x = box.x + box.w / 2;
blob3.y = box.y + box.h / 2;
```

Why this fixes the teleport bug

Sometimes (especially when panic/wobble changes size slightly), the blob ends a frame **1–2 pixels inside the floor**.

Next frame:

- horizontal collision sees an overlap with the *floor rectangle*
- treats it like a wall
- snaps the blob left → teleport

This line guarantees:

"At the end of every frame, the blob can *never* still be inside the floor."

TL;DR

Add **exactly one line**, **exactly here**:

```
// AFTER vertical collisions
box.y = min(box.y, floorY3 - box.h);
```

You're doing great — this is a classic platformer edge case, not a mistake on your part. If it *still* happens after this, I can give you the "industrial-strength" fix, but this should solve it cleanly.

its looking great now but there is a glitch with jumping on the platform. I found that i was able to jump on the rightmost platform but i cant seem to get on the leftmost one. When i do get above it i glitch and come off of it
Thought for 10s

Yep — this symptom makes a lot of sense with your current "fear" mechanics.

Why it happens

Two things are teaming up:

1. **Your collision size changes while you play**
   You're using a collision radius that depends on `wobble`, and `wobble` changes with `panic` every frame. When the hitbox size changes while you're *standing* on a platform, the collision box can suddenly overlap the platform and get "resolved" in a way that nudges you off.
2. **Your tremble adds real sideways velocity while grounded**
   So even if you land cleanly, the tremble can gently push you off the edge of a platform (especially the one that sits in darker space where panic is higher).

Rightmost platform is easier because (in your lighting layout) it's likely closer to a light pool → lower panic → less wobble change + less tremble.

Fix it cleanly (2 small edits)

✅ Fix 1: Make the physics hitbox a constant size

Keep wobble/panic purely visual. Collisions should use a stable radius.

In `draw()`, find where you currently compute collision radius (you'll see something like `collisionR = blob3.r + ...`).

Replace it with:

```
// Physics hitbox should be stable (don't tie it to wobble/panic)
let collisionR = blob3.r; // or blob3.r + 1 if you want it slightly forgiving
```

That alone will stop a lot of the "glitch off platform" behavior.

✅ Fix 2: Only tremble when you're NOT actively moving

This prevents fear-jitter from physically pushing you off platforms.

Find your tremble block:

```
if (blob3.onGround && panic > 0.15) {
  blob3.vx += (noise(frameCount * 0.25) - 0.5) * 0.28 * panic;
}
```

Replace it with:

```
// Tremble only when standing still (so it doesn't shove you off ledges)
if (blob3.onGround && panic > 0.15 && move === 0) {
  blob3.vx += (noise(frameCount * 0.25) - 0.5) * 0.18 * panic;
}

// Tiny "stick" so micro-velocities don't slide you off platforms
if (blob3.onGround && abs(blob3.vx) < 0.05) blob3.vx = 0;
```

What you should see after this

- You can land on the left platform consistently
- You don't "buzz" off the platform when fear is high
- Fear still reads because wobble/breath speed + lighting are unchanged — we just stopped fear from breaking platforming

---

If you paste the exact snippet around your `collisionR` and the tremble section, I can tell you the *exact lines* to replace in your current file so you don't have to hunt.