Puget Sound Sonic Map — Brief v5

Targeted fixes only. Do not touch files not mentioned. Run `npm run dev` after each section. Final `npm run build`.

FIX 1 — PANNING: DRAG CONFLICT WITH DISPLACEMENT

The pan drag feels wrong because p5's mouseDragged fires simultaneously with the displacement field calculation, creating a tug-of-war.

In SonicMap.jsx, two changes:

Change A — In `p.mouseDragged`, set a dragging flag:

```js
p.mousePressed = () => {
  s.isDragging = true;
  s.dragStartX = p.mouseX - s.panX;
  s.dragStartY = p.mouseY - s.panY;
}
p.mouseReleased = () => {
  s.isDragging = false;
}
p.mouseDragged = () => {
  if (!s.isDragging) return;
  s.panX = p.mouseX - s.dragStartX;
  s.panY = p.mouseY - s.dragStartY;
  s.panX = Math.max(-(p.width * 0.4), Math.min(p.width * 0.4, s.panX));
  s.panY = Math.max(-(p.height * 0.4), Math.min(p.height * 0.4, s.panY));
}
```

Change B — In `buildField()`, skip mouse displacement entirely when dragging:

```js
```

```
// Replace the existing mouse displacement block with:
if (!s.isDragging) {
  const mdx = nx - mxN;
  const mdy = ny - myN;
  const mDist = Math.sqrt(mdx*mdx + mdy*mdy);
  const mInfluence = Math.max(0, 1 - mDist / 0.22);
  const mSoft = mInfluence * mInfluence * (3 - 2 * mInfluence);
  const speed = Math.sqrt((s.vx||0)**2 + (s.vy||0)**2);
  const velocityBoost = Math.min(speed * 0.006, 0.35);
  val -= mSoft * (0.65 + velocityBoost);
}
```

Change C — Clear the ghost layer on pan to prevent smearing:

```
js

// In p.draw(), after updating panX/panY:
const panDelta = Math.abs((s.panX - (s.lastPanX||0))) + Math.abs((s.panY - (s.lastPanY||0)));
if (panDelta > 2) {
  ghostLayer.background(0, 3, 13); // clear trails when panning
}
s.lastPanX = s.panX;
s.lastPanY = s.panY;
```

Change D — Update cursor CSS to reflect drag state. In Cursor.jsx, accept `isDragging` prop. When true: `cursor: 'grabbing'`, hide the crosshair lines, show only dot + ring at 36px. Pass isDragging from SonicMap via a `onDragStateChange(bool)` callback to App, store in a ref, pass down to Cursor.

FIX 2 — BIOLUMINESCENT GLOW: MORE VISIBLE

The glow is too subtle except for green. Two adjustments:

In SonicMap.jsx, in the glow pass inside drawContours():

Increase max alpha from `prox * 60` to `prox * 110`:

```
js

const glowAlpha = cellProx * cellProx * prox * 110; // was 60
```

Increase stroke weight for glow lines from `0.8` to `1.4`:

```
js
```

```
pg.strokeWeight(1.4); // was 0.8
```

Add a second glow pass — draw the same lines again at half alpha but double blur effect using the canvas drawingContext: After the main contour+glow draw on `contourLayer`, add:

```
js

// In p.draw(), after `p.image(contourLayer, 0, 0)`:
// Draw an extra blurred copy of contourLayer for bloom boost
p.drawingContext.globalAlpha = 0.25; // was 0.18
p.drawingContext.filter = 'blur(6px)'; // was 4px
p.image(contourLayer, 0, 0);
p.drawingContext.filter = 'blur(16px)'; // was 12px
p.drawingContext.globalAlpha = 0.12;   // was 0.07
p.image(contourLayer, 0, 0);
p.drawingContext.filter = 'none';
p.drawingContext.globalAlpha = 1.0;
```

FIX 3 — BOTTOM-RIGHT SPECIES TEXT (replace tooltip)

Remove SpeciesTooltip.jsx entirely.
Replace with a single line of text, same style as the old bottom-left label,
but positioned bottom-right.

Delete src/components/SpeciesTooltip.jsx

In HUD.jsx, add this element:

```
jsx
```

```
{/* Species label — bottom right */}
<div style={{
  position: 'fixed',
  bottom: 28,
  right: 32,
  zIndex: 50,
  fontFamily: "'DM Mono', 'Courier New', monospace",
  fontSize: 10,
  letterSpacing: '0.15em',
  textTransform: 'uppercase',
  color: speciesPing
    ? `rgba(${speciesPing.color[0]}, ${speciesPing.color[1]}, ${speciesPing.color[2]}, 0.9)`
    : 'rgba(0,180,220,0)',
  textShadow: speciesPing
    ? `0 0 20px rgba(${speciesPing.color[0]}, ${speciesPing.color[1]}, ${speciesPing.color[2]}, 0.5)`
    : 'none',
  transition: 'color 0.3s ease, text-shadow 0.3s ease',
  pointerEvents: 'none',
}}>
  {speciesPing
    ? `${speciesPing.name} — ${speciesPing.latin}`
    : '\u200b' /* zero-width space keeps layout stable */
  }
</div>
```

Update HUD props to accept `speciesPing` (the full species object or null). The label color uses the species' own color — so orca label glows blue, eelgrass label glows green, etc.

In App.jsx:

- Pass `speciesPing={speciesPing}` to HUD
- Remove all SpeciesTooltip imports and JSX
- Remove cursorX/Y tracking if it was only used for tooltip positioning

---

FIX 4 — AUDIO: DEPTH-MAPPED AMBIENT ZONES + LIGHTER BASE DRONE

This fixes two things: the zone ambient system (which may not be playing continuously as intended) and the base drone (too dark).

4A. Base drone — shift from dark to ambiguous

In AudioEngine.js, change the drone chord:

```
// OLD — heavy D minor
drone.triggerAttack(['D1', 'A1', 'D2', 'F2']);
subDrone.triggerAttack('D0');

// NEW — open fifth + second, no minor third = ambiguous, not dark
drone.triggerAttack(['D1', 'A1', 'E2', 'B2']);
subDrone.triggerAttack('A0'); // root shift to A — warmer, less ominous
```

Also reduce drone volume slightly: change PolySynth volume from `-20` to `-23`, but increase subDrone from `-24` to `-20` — more warmth in the sub, less harshness in the mid drone.

Add a slow Tone.AutoFilter on the drone for organic movement:

```
const droneFilter = new Tone.AutoFilter({
  frequency: 0.08,  // very slow
  baseFrequency: 200,
  octaves: 2,
  wet: 0.4
}).start();
drone.connect(droneFilter);
droneFilter.connect(this.masterFilter);
// (disconnect drone's direct connection to masterFilter)
```

4B. Zone layers — depth-mapped pitch registers

The zone layer system should already exist from v4. If it's not playing continuous tones when cursor enters a species radius, the issue is likely that `updateZones` is not being called every frame, or the `triggerAttack` is firing but the gain isn't ramping up.

Debug check first: Add a `console.log('zone active:', id, proximity)` inside the `proximity > 0.05` branch of `updateZones`. Confirm it fires. If it does but no sound: the gain node isn't connected correctly. If it doesn't fire: `updateZones` isn't being called from the render loop.

Fix the zone layer gain connection — ensure this chain is correct:

```
synth → gainNode (Tone.Volume, starts at -60) → masterFilter → chorus → reverb → destination
```

Verify in `_createZoneLayer`: `gain.connect(this.masterFilter)` not `.toDestination()`.

Depth-mapped pitch: Each species has a `depth` value (0–1, where 1 = deepest). Use this to transpose the zone layer note:

```
js

// In _createZoneLayer, after getting `def`:
// Map species depth to octave offset
// depth 0.0–0.3 (shallow: eelgrass, kelp) → +1 octave (brighter)
// depth 0.3–0.7 (mid: salmon, crab, seal) → 0 octave (base register)
// depth 0.7–1.0 (deep: orca, octopus)    → -1 octave (darker)
const octaveOffset = species.depth < 0.3 ? 1 : species.depth > 0.7 ? -1 : 0;

// Apply to note in triggerAttack:
// Parse the note letter and number, add octaveOffset to the number
function transposeNote(note, offset) {
  const letter = note.replace(/[0-9]/g, '');
  const octave = parseInt(note.replace(/[^0-9]/g, '')) + offset;
  return `${letter}${Math.max(0, Math.min(8, octave))}`;
}

// In updateZones, when triggering:
const transposedNote = transposeNote(def.notes[0], octaveOffset);
layer.synth.triggerAttack(transposedNote);
```

Volume by depth — shallow species get slightly higher volume (they're in the water column we inhabit, not the abyss):

```
js

// In _createZoneLayer, adjust def.volume:
const depthVolumeOffset = (1 - species.depth) * 4; // +4dB at surface, 0 at depth
// Use (def.volume + depthVolumeOffset) as the target volume
```

4C. Ensure updateZones is called every frame

In App.jsx or SonicMap.jsx, confirm `onProximityUpdate` / `handleProximityUpdate` is called inside `p.draw()` on every frame, not just on species change.

The call should look like:

```
js

// Inside p.draw(), every frame:
onProximityUpdate(proximityMap); // proximityMap built fresh each frame
```

If this callback causes React re-renders on every frame (60fps state updates), switch to a ref-based approach:

```js
// Instead of setState, use a ref that AudioEngine reads directly
const proximityRef = useRef({});
// In p.draw callback: proximityRef.current = proximityMap;
// AudioEngine.updateZones reads from proximityRef every 100ms via setInterval
```

FIX 5 — ENTRY SCREEN: TOPOLOGY CURSOR PORTAL

Replace the static SVG decorative element on the right panel of IntroScreen.jsx with a live mini p5 sketch that shows the marching squares topology, centered around the cursor position. When you click Enter, it expands and morphs into the main canvas.

5A. Create src/components/TopologyPortal.jsx

```jsx
```

```jsx
import { useEffect, useRef } from 'react';

// A self-contained p5 sketch for the entry screen right panel.
// Shows a small marching-squares topographic field that follows the cursor.
// Clicking triggers an expand animation then calls onEnter().

export default function TopologyPortal({ onEnter }) {
  const containerRef = useRef(null);
  const expandingRef = useRef(false);

  useEffect(() => {
    let p5instance;
    import('p5').then(({ default: P5 }) => {
      const sketch = (p) => {
        // Smaller grid — this is a preview, not the full map
        const COLS = 32, ROWS = 22;
        let mx = 0, my = 0;
        let expandT = 0; // 0 = normal, 1 = fully expanded
        let ghostLayer;

        p.setup = () => {
          const cnv = p.createCanvas(
            containerRef.current.offsetWidth,
            containerRef.current.offsetHeight
          );
          cnv.parent(containerRef.current);
          ghostLayer = p.createGraphics(p.width, p.height);
          ghostLayer.background(0, 3, 13, 0);
          p.frameRate(60);
        };

        p.mouseMoved = () => {
          // Convert to canvas-local coordinates
          const rect = containerRef.current.getBoundingClientRect();
          mx = p.mouseX;
          my = p.mouseY;
        };

        p.draw = () => {
          const W = p.width, H = p.height;
          const cellW = W / COLS, cellH = H / ROWS;
          const t = p.frameCount * 0.012;
```

```
// If expanding: grow expandT toward 1
if (expandingRef.current && expandT < 1) {
  expandT = Math.min(1, expandT + 0.03);
}

// Build field — same logic as main map but cursor-centered
const field = [];
for (let r = 0; r <= ROWS; r++) {
  field[r] = [];
  for (let c = 0; c <= COLS; c++) {
    const nx = c / COLS;
    const ny = r / ROWS;
    let val = p.noise(nx * 2.5 + t * 0.04, ny * 2.0 + t * 0.03, t * 0.01);

    // Mouse depression
    const mxN = mx / W;
    const myN = my / H;
    const mdx = nx - mxN, mdy = ny - myN;
    const mDist = Math.sqrt(mdx*mdx + mdy*mdy);
    const mSoft = Math.max(0, 1 - mDist / 0.25) ** 2;
    val -= mSoft * 0.5;

    // Expand effect: as expandT grows, push topology outward from center
    if (expandT > 0) {
      const cx = nx - 0.5, cy = ny - 0.5;
      const cDist = Math.sqrt(cx*cx + cy*cy);
      val -= expandT * (1 - cDist) * 0.8;
    }

    field[r][c] = Math.max(0, Math.min(1, val));
  }
}

// Ghost fade
ghostLayer.noStroke();
ghostLayer.fill(0, 3, 13, expandingRef.current ? 30 : 18);
ghostLayer.rect(0, 0, W, H);
p.image(ghostLayer, 0, 0);

// Draw contours
const LEVELS = 10;
for (let level = 0; level < LEVELS; level++) {
  const iso = 0.15 + (level / LEVELS) * 0.65;
  const lv = level / LEVELS;
```

```
      // Color shifts toward white/bright as expandT grows (portal opening)
      const alpha = p.lerp(p.lerp(30, 100, lv), 200, expandT);
      p.stroke(
        p.lerp(0,  p.lerp(20,  180, expandT), lv),
        p.lerp(60, p.lerp(180, 220, expandT), lv),
        p.lerp(140,p.lerp(255, 255, expandT), lv),
        alpha
      );
      p.strokeWeight(p.lerp(0.3, 0.8, lv));
      p.noFill();

      for (let r = 0; r < ROWS; r++) {
        for (let c = 0; c < COLS; c++) {
          const x0 = c * cellW, y0 = r * cellH;
          const v00=field[r][c], v10=field[r][c+1];
          const v01=field[r+1][c], v11=field[r+1][c+1];
          const idx=(v00>iso?8:0)|(v10>iso?4:0)|(v11>iso?2:0)|(v01>iso?1:0);
          if (idx===0||idx===15) continue;
          const it=(a,b,v)=>(iso-a)/(b-a)*v;
          const top   ={x:x0+it(v00,v10,cellW),y:y0};
          const bottom={x:x0+it(v01,v11,cellW),y:y0+cellH};
          const left  ={x:x0,                  y:y0+it(v00,v01,cellH)};
          const right ={x:x0+cellW,            y:y0+it(v10,v11,cellH)};
          const lm={1:[[left,bottom]],2:[[right,bottom]],3:[[left,right]],
            4:[[top,right]],5:[[top,left],[right,bottom]],6:[[top,bottom]],
            7:[[top,left]],8:[[top,left]],9:[[top,bottom]],
            10:[[top,right],[left,bottom]],11:[[top,right]],
            12:[[left,right]],13:[[right,bottom]],14:[[left,bottom]]};
          const segs=lm[idx]; if(!segs) continue;
          for(const[a,b2]of segs) p.line(a.x,a.y,b2.x,b2.y);
        }
      }

      // If fully expanded, call onEnter
      if (expandT >= 1) {
        p5instance?.remove();
        onEnter();
      }
    };
  };
  p5instance = new P5(sketch);
});
return () => p5instance?.remove();
```

```
  }, [onEnter]);

  const handleEnterClick = () => {
    expandingRef.current = true;
  };

  return (
    <div style={{ position: 'relative', width: '100%', height: '100%' }}>
      {/* The live topology canvas */}
      <div ref={containerRef} style={{ position: 'absolute', inset: 0 }} />

      {/* Enter text — positioned bottom-right of this panel */}
      <div
        onClick={handleEnterClick}
        style={{
          position: 'absolute',
          bottom: 48,
          right: 52,
          fontFamily: "'DM Mono', monospace",
          fontSize: 10,
          letterSpacing: '0.25em',
          textTransform: 'uppercase',
          color: 'rgba(0, 180, 220, 0.5)',
          cursor: 'pointer',
          transition: 'color 0.3s ease',
          zIndex: 10,
          userSelect: 'none',
        }}
        onMouseEnter={e => e.currentTarget.style.color = 'rgba(0, 210, 255, 0.95)'}
        onMouseLeave={e => e.currentTarget.style.color = 'rgba(0, 180, 220, 0.5)'}
      >
        Enter ↓
      </div>
    </div>
  );
}
```

5B. Update IntroScreen.jsx

Replace the right panel's current content (the static SVG / decorative element) with `<TopologyPortal onEnter={handleClick} />`.

The right panel div should have `overflow: hidden` and `position: relative` so the canvas is clipped to the panel bounds.

Remove the old "Enter ↓" button from the left panel entirely —
it now lives inside TopologyPortal on the right.

The left panel keeps: coordinates label, "Puget Sound" title,
"A Sonic Field Study" subtitle, and the body text. No button.

```jsx
// IntroScreen.jsx right panel:
<div style={{
  width: '60%',
  height: '100%',
  position: 'relative',
  overflow: 'hidden',
}}>
  <TopologyPortal onEnter={handleClick} />
</div>
```

EXECUTION ORDER

1. Fix 1 (panning) — test drag feels smooth, no topology fighting
2. Fix 2 (glow brightness) — visual only, quick
3. Fix 3 (species text bottom-right) — removes tooltip, adds HUD label
4. Fix 4A (drone lightening) — audio only, test sound
5. Fix 4B+4C (zone layers debug + depth mapping) — most complex, test carefully
6. Fix 5 (entry screen topology) — test the expand animation timing

KNOWN LIKELY ISSUES

- Fix 4C performance: if calling onProximityUpdate 60× per second causes jank, use the ref+interval approach described. 100ms polling is fine for audio (human ear threshold for zone transitions is ~200ms).
- Fix 5 expand timing: if `expandT >= 1` fires too fast or too slow, adjust the increment: `expandT + 0.03` = ~1.1 seconds to expand. Slower = `0.015` (~2s). Faster = `0.05` (~0.7s).
- Fix 5 canvas sizing: TopologyPortal uses `containerRef.current.offsetWidth` in setup. If this returns 0 (layout not yet painted), add a `setTimeout(() => p5instance = new P5(sketch), 50)` delay.

Report all judgment calls.