Puget Sound Sonic Map — Refinement Brief v4

Everything is running cleanly. This is a focused refinement pass. Work through each part in order. Run `npm run dev` between parts to catch errors early. Final: `npm run build` — zero errors required.

PART 1 — SPECIES DATA: VERIFIED & EXPANDED

Replace src/data/MarineData.json entirely with this verified dataset.
All species confirmed present in Puget Sound / Salish Sea via WDFW, NOAA, and iNaturalist.
Diamond colors are now ecologically coherent — see color logic notes.

COLOR LOGIC:

- Mammals (cetaceans): `#00b4ff` — deep ocean blue
- Mammals (pinnipeds): `#4fc3f7` — lighter blue
- Cephalopods: `#b44fff` — deep violet
- Plants (eelgrass): `#00e676` — bright green
- Plants (kelp): `#69f0ae` — softer green
- Fish (salmon): `#ff5252` — urgent red (endangered)
- Fish (herring): `#ff8a65` — warm coral
- Crustaceans: `#ffb300` — amber
- Birds: `#e0e0e0` — pale silver (they hunt the water)

```json
```

```
{
  "bounds": {
    "latMin": 47.0, "latMax": 48.8,
    "lonMin": -123.2, "lonMax": -122.0,
    "description": "Puget Sound / Salish Sea, Washington State"
  },
  "species": [
    {
      "id": "orca-j-pod",
      "name": "Orca",
      "latin": "Orcinus orca",
      "commonName": "Southern Resident Killer Whale",
      "category": "mammal",
      "status": "Critically Endangered",
      "depthRange": "0–300m, surface-active",
      "source": "NOAA Fisheries ESA listing 2005; SRKW Recovery Plan",
      "x": 0.25, "y": 0.12,
      "radius": 0.09, "depth": 0.88,
      "color": [0, 180, 255],
      "fact": "Only 73 individuals remain. They communicate in dialects unique to each pod, passed between generations."
    },
    {
      "id": "orca-l-pod",
      "name": "Orca",
      "latin": "Orcinus orca",
      "commonName": "Southern Resident Killer Whale — L Pod",
      "category": "mammal",
      "status": "Critically Endangered",
      "depthRange": "0–300m, surface-active",
      "source": "NOAA SRKW monitoring; Strait of Juan de Fuca sightings database",
      "x": 0.55, "y": 0.08,
      "radius": 0.08, "depth": 0.85,
      "color": [0, 180, 255],
      "fact": "L Pod is the largest resident pod. Members sometimes range to the outer Pacific coast."
    },
    {
      "id": "harbor-porpoise",
      "name": "Harbor Porpoise",
      "latin": "Phocoena phocoena",
      "commonName": "Harbor Porpoise",
      "category": "mammal",
      "status": "Least Concern",
      "depthRange": "0–200m, prefers channels",
```

```
  "source": "Cascadia Research; WDFW Marine Mammal Investigations",
  "x": 0.48, "y": 0.45,
  "radius": 0.06, "depth": 0.62,
  "color": [79, 195, 247],
  "fact": "Echolocates at 130 kHz — among the highest frequencies of any cetacean. Rarely seen; more often heard."
},
{
  "id": "harbor-seal",
  "name": "Harbor Seal",
  "latin": "Phoca vitulina",
  "commonName": "Harbor Seal",
  "category": "mammal",
  "status": "Least Concern",
  "depthRange": "Intertidal to 300m",
  "source": "WDFW Pinniped Haul-out Survey; Hood Canal monitoring program",
  "x": 0.12, "y": 0.55,
  "radius": 0.065, "depth": 0.55,
  "color": [79, 195, 247],
  "fact": "~11,000 in Puget Sound. Can dive to 300m and hold breath for 30 minutes. Pups born on beaches in June."
},
{
  "id": "steller-sea-lion",
  "name": "Steller Sea Lion",
  "latin": "Eumetopias jubatus",
  "commonName": "Steller Sea Lion",
  "category": "mammal",
  "status": "Near Threatened",
  "depthRange": "Surface to 180m",
  "source": "NOAA Fisheries; seasonal haul-out sites at Protection Island",
  "x": 0.70, "y": 0.10,
  "radius": 0.055, "depth": 0.50,
  "color": [79, 195, 247],
  "fact": "Winter visitors from Alaska. Males can reach 1,100 kg. Roost on navigation buoys and rock outcroppings."
},
{
  "id": "giant-pacific-octopus",
  "name": "Giant Pacific Octopus",
  "latin": "Enteroctopus dofleini",
  "commonName": "Giant Pacific Octopus",
  "category": "cephalopod",
  "status": "Not Evaluated",
  "depthRange": "10–150m, rocky substrate",
  "source": "WDFW dive surveys; Puget Sound Nearshore Ecosystem Restoration Project",
  "x": 0.38, "y": 0.72,
```

```
  "radius": 0.08, "depth": 0.92,
  "color": [180, 40, 255],
  "fact": "World's largest octopus — arm span to 4.3m. Lives in dens, hunts at night. Lifespan: only 3–5 years."
 },
 {
  "id": "market-squid",
  "name": "Market Squid",
  "latin": "Doryteuthis opalescens",
  "commonName": "Opalescent Inshore Squid",
  "category": "cephalopod",
  "status": "Not Evaluated",
  "depthRange": "0–200m, mid-water",
  "source": "WDFW spawn surveys; Puget Sound mid-channel trawl data",
  "x": 0.62, "y": 0.38,
  "radius": 0.07, "depth": 0.78,
  "color": [180, 40, 255],
  "fact": "Schools of thousands bioluminesce blue-green during spawning. Each individual lives less than one year."
 },
 {
  "id": "eelgrass-padilla",
  "name": "Eelgrass",
  "latin": "Zostera marina",
  "commonName": "Common Eelgrass — Padilla Bay",
  "category": "plant",
  "status": "Declining",
  "depthRange": "Intertidal to 6m",
  "source": "NERR Padilla Bay; WDFW SAV mapping 2022 — 8,000 acres documented",
  "x": 0.18, "y": 0.10,
  "radius": 0.13, "depth": 0.22,
  "color": [0, 230, 118],
  "fact": "Padilla Bay holds the largest eelgrass bed in Washington — 8,000 acres. Nursery for juvenile Chinook salmon."
 },
 {
  "id": "eelgrass-nisqually",
  "name": "Eelgrass",
  "latin": "Zostera marina",
  "commonName": "Common Eelgrass — Nisqually Delta",
  "category": "plant",
  "status": "Declining",
  "depthRange": "Intertidal to 6m",
  "source": "WDFW SAV; Nisqually Delta restoration monitoring 2009–2023",
  "x": 0.45, "y": 0.88,
  "radius": 0.10, "depth": 0.18,
  "color": [0, 230, 118],
```

```
  "fact": "Each acre sequesters carbon at 35× the rate of a tropical rainforest. The Nisqually bed was restored from farmland
 },
 {
  "id": "bull-kelp-sanjuans",
  "name": "Bull Kelp",
  "latin": "Nereocystis luetkeana",
  "commonName": "Bull Kelp — San Juan Islands",
  "category": "plant",
  "status": "Declining",
  "depthRange": "5–20m, rocky substrate",
  "source": "WDFW Kelp Canopy Survey 2022; MRC of the Salish Sea",
  "x": 0.30, "y": 0.06,
  "radius": 0.10, "depth": 0.32,
  "color": [105, 240, 174],
  "fact": "Grows up to 10cm per day. Canopy forests shelter 800+ species. San Juan beds are among the densest remaining."
 },
 {
  "id": "bull-kelp-hoodcanal",
  "name": "Bull Kelp",
  "latin": "Nereocystis luetkeana",
  "commonName": "Bull Kelp — Hood Canal",
  "category": "plant",
  "status": "Declining",
  "depthRange": "5–20m, rocky substrate",
  "source": "WDFW Kelp Survey: Hood Canal 40% canopy loss 1990–2022",
  "x": 0.10, "y": 0.62,
  "radius": 0.09, "depth": 0.28,
  "color": [105, 240, 174],
  "fact": "Hood Canal has lost 40% of its kelp canopy since 1990. Warming waters and low-oxygen events are the primary c
 },
 {
  "id": "chinook-salmon",
  "name": "Chinook Salmon",
  "latin": "Oncorhynchus tshawytscha",
  "commonName": "King Salmon — Puget Sound ESU",
  "category": "fish",
  "status": "Threatened",
  "depthRange": "0–60m, migratory",
  "source": "NOAA ESA listing 1999; Puget Sound Salmon Recovery Plan",
  "x": 0.50, "y": 0.50,
  "radius": 0.08, "depth": 0.70,
  "color": [255, 82, 82],
  "fact": "Primary prey of Southern Resident Orcas. Their decline drives orca starvation. Listed under ESA since 1999."
 },
```

```
{
  "id": "coho-salmon",
  "name": "Coho Salmon",
  "latin": "Oncorhynchus kisutch",
  "commonName": "Silver Salmon",
  "category": "fish",
  "status": "Threatened",
  "depthRange": "0–40m, nearshore and river mouths",
  "source": "NOAA ESA listing; Puget Sound coho ESU recovery plan",
  "x": 0.35, "y": 0.58,
  "radius": 0.07, "depth": 0.65,
  "color": [255, 82, 82],
  "fact": "Urban tire rubber chemical 6PPD-quinone is lethal to coho within hours of rain runoff. A leading cause of collaps
},
{
  "id": "pacific-herring",
  "name": "Pacific Herring",
  "latin": "Clupea pallasii",
  "commonName": "Pacific Herring — Cherry Point",
  "category": "fish",
  "status": "Declining",
  "depthRange": "0–250m, schooling",
  "source": "WDFW Spawn Survey: Cherry Point largest US West Coast aggregation",
  "x": 0.82, "y": 0.15,
  "radius": 0.075, "depth": 0.45,
  "color": [255, 138, 101],
  "fact": "Cherry Point aggregation is the largest Pacific herring spawn on the US West Coast. Base of the entire Salish Sea
},
{
  "id": "dungeness-crab",
  "name": "Dungeness Crab",
  "latin": "Metacarcinus magister",
  "commonName": "Dungeness Crab",
  "category": "crustacean",
  "status": "Stable",
  "depthRange": "0–180m, sand and mud",
  "source": "WDFW Crab Survey; Puget Sound subtidal monitoring",
  "x": 0.58, "y": 0.52,
  "radius": 0.07, "depth": 0.48,
  "color": [255, 179, 0],
  "fact": "Migrate seasonally between shallow spawning grounds and deep overwintering sites. Molt annually, leaving ghos
},
{
  "id": "geoduck",
```

```
    "name": "Geoduck",
    "latin": "Panopea generosa",
    "commonName": "Pacific Geoduck Clam",
    "category": "crustacean",
    "status": "Stable",
    "depthRange": "Intertidal to 110m, buried in substrate",
    "source": "WDFW Geoduck Survey; Puget Sound aquaculture assessment",
    "x": 0.32, "y": 0.72,
    "radius": 0.065, "depth": 0.38,
    "color": [255, 179, 0],
    "fact": "Can live 168 years — the oldest found was 179. Buried up to 1m deep. Filter feeders critical to water clarity."
  }
 ]
}
```

PART 2 — AUDIO: PROXIMITY LAYERS REPLACING PINGS

This is a significant audio architecture change. Read fully before implementing.

Concept

Each species zone now has TWO audio behaviors:

1. **Zone layer** — a held musical texture that fades in as you enter the species radius, fades out as you leave. This creates a generative score from overlapping zones.
2. **Diamond ping** — a single sharp attack note only when cursor is within 20px of the exact diamond center. One ping per visit, 3-second cooldown.

2A. Rewrite AudioEngine.js

```
js
```

```
export default class AudioEngine {
  constructor() {
    this.started = false;
    this.Tone = null;
    this.masterFilter = null;
    this.reverb = null;
    this.zoneLayers = {};      // { speciesId: { synth, gain, active } }
    this.activeZones = new Set();
    this.lastPingTime = {};
  }

  async start() {
    if (this.started) return;
    const Tone = await import('tone');
    await Tone.start();
    this.Tone = Tone;
    this.started = true;

    // Master chain: filter → reverb → destination
    this.reverb = new Tone.Reverb({ decay: 14, wet: 0.65 }).toDestination();
    const chorus = new Tone.Chorus({ frequency: 0.2, delayTime: 4, depth: 0.5, wet: 0.3 });
    chorus.start();
    chorus.connect(this.reverb);

    this.masterFilter = new Tone.Filter({ frequency: 800, type: 'lowpass', rolloff: -24 });
    this.masterFilter.connect(chorus);

    // Deep drone (unchanged)
    const drone = new Tone.PolySynth(Tone.Synth, {
      oscillator: { type: 'sawtooth' },
      envelope: { attack: 3, decay: 2, sustain: 0.8, release: 6 },
      volume: -20
    });
    drone.connect(this.masterFilter);

    const subDrone = new Tone.Synth({
      oscillator: { type: 'sine' },
      envelope: { attack: 4, decay: 3, sustain: 1, release: 8 },
      volume: -24   // increased from -28
    });
    subDrone.connect(this.masterFilter);

    const lfo = new Tone.LFO({ frequency: 0.05, min: -6, max: 6 }).start();
```

```
lfo.connect(drone.detune);

drone.triggerAttack(['D1', 'A1', 'D2', 'F2']);
subDrone.triggerAttack('D0');
this.drone = drone;

// Zone layer definitions per category
// These are the held tones that fade in/out with proximity
this.zoneLayerDefs = {
  mammal: {
    notes: ['D2', 'A2', 'F2', 'C3'],
    type: 'sine',
    volume: -16,   // boosted — low frequencies need more gain
    attack: 2.5, release: 4.0,
    filterFreq: 180,
    description: 'low resonant whale-like tone'
  },
  cephalopod: {
    notes: ['A3', 'D4', 'G4', 'C4'],
    type: 'fmsine',
    modulationIndex: 5,
    volume: -19,
    attack: 1.5, release: 3.0,
    filterFreq: 600,
    description: 'eerie mid shimmer'
  },
  plant: {
    notes: ['G5', 'A5', 'D6', 'E5'],
    type: 'triangle',
    volume: -21,
    attack: 2.0, release: 3.5,
    filterFreq: 1800,
    description: 'high shimmer'
  },
  crustacean: {
    notes: ['E3', 'G3', 'B3', 'D4'],
    type: 'square',
    partialCount: 2,
    volume: -18,   // boosted — was too quiet
    attack: 1.8, release: 2.8,
    filterFreq: 400,
    description: 'rhythmic mid-low'
  },
  fish: {
```

```
    notes: ['B3', 'E4', 'G4', 'D4'],
    type: 'sine',
    volume: -17,   // boosted
    attack: 2.0, release: 3.5,
    filterFreq: 500,
    description: 'flowing mid tone'
  }
};

// Diamond ping synths (sharp attack, long decay — one per category)
this.pingSynths = {
  mammal: new Tone.Synth({
    oscillator: { type: 'sine' },
    envelope: { attack: 0.02, decay: 0.4, sustain: 0.05, release: 4.5 },
    volume: -12   // boosted significantly
  }).connect(this.reverb),

  cephalopod: new Tone.Synth({
    oscillator: { type: 'fmsine', modulationIndex: 8 },
    envelope: { attack: 0.03, decay: 0.6, sustain: 0.0, release: 3.5 },
    volume: -15
  }).connect(this.reverb),

  plant: new Tone.Synth({
    oscillator: { type: 'triangle' },
    envelope: { attack: 0.01, decay: 0.3, sustain: 0.0, release: 3.0 },
    volume: -14
  }).connect(this.reverb),

  crustacean: new Tone.Synth({
    oscillator: { type: 'square', partialCount: 3 },
    envelope: { attack: 0.01, decay: 0.2, sustain: 0.0, release: 2.5 },
    volume: -14   // boosted from -22
  }).connect(this.reverb),

  fish: new Tone.Synth({
    oscillator: { type: 'sine' },
    envelope: { attack: 0.05, decay: 0.8, sustain: 0.1, release: 5.0 },
    volume: -13   // boosted
  }).connect(this.reverb)
};

// Ping pitch map
this.pingPitches = {
```

```
    mammal:     ['D2', 'A2', 'D3', 'F3'],
    cephalopod: ['A3', 'C4', 'E4', 'G4'],
    plant:      ['D5', 'F5', 'A5', 'C6'],
    crustacean: ['G3', 'B3', 'D4', 'F4'],
    fish:       ['E3', 'G3', 'B3', 'D4', 'E4']
  };
}

// Called every frame with array of { species, proximity (0-1) }
// proximity = 0 means outside zone, 1 = center
updateZones(proximityMap) {
  if (!this.started) return;
  const Tone = this.Tone;

  for (const [id, { species, proximity }] of Object.entries(proximityMap)) {
    if (proximity > 0.05) {
      // Species is in range — ensure zone layer exists and is playing
      if (!this.zoneLayers[id]) {
        this._createZoneLayer(species);
      }
      const layer = this.zoneLayers[id];
      if (!layer) continue;

      // Smooth gain change based on proximity
      // Use a curve: gain ramps from silence to full over proximity 0.05→0.5
      const targetGain = Math.min(1, proximity * 2);
      const def = this.zoneLayerDefs[species.category];
      const dbTarget = def ? (def.volume + (1 - targetGain) * -20) : -30;
      layer.gain.rampTo(dbTarget, 0.8);

      if (!this.activeZones.has(id)) {
        this.activeZones.add(id);
        layer.synth.triggerAttack(def.notes[0]);
      }
    } else {
      // Species out of range — fade out
      if (this.activeZones.has(id) && this.zoneLayers[id]) {
        this.activeZones.delete(id);
        const layer = this.zoneLayers[id];
        layer.gain.rampTo(-60, 3.0);
        // Release after fade
        setTimeout(() => {
          try { layer.synth.triggerRelease(); } catch(e) {}
        }, 3200);
```

```
      }
    }
  }
}

_createZoneLayer(species) {
  if (this.zoneLayers[species.id]) return;
  const Tone = this.Tone;
  const def = this.zoneLayerDefs[species.category];
  if (!def) return;

  try {
    const gain = new Tone.Volume(-60); // start silent
    gain.connect(this.masterFilter);

    const opts = {
      oscillator: { type: def.type },
      envelope: {
        attack: def.attack, decay: 1.0,
        sustain: 0.8, release: def.release
      },
      volume: 0
    };
    if (def.modulationIndex) opts.oscillator.modulationIndex = def.modulationIndex;
    if (def.partialCount) opts.oscillator.partialCount = def.partialCount;

    const synth = new Tone.Synth(opts);
    synth.connect(gain);

    this.zoneLayers[species.id] = { synth, gain };
  } catch(e) {
    console.warn('Zone layer creation failed:', e);
  }
}

// Sharp ping — only when cursor is very close to diamond center
triggerDiamondPing(species) {
  if (!this.started) return;
  const now = Date.now();
  if (this.lastPingTime[species.id] && now - this.lastPingTime[species.id] < 3000) return;
  this.lastPingTime[species.id] = now;

  const synth = this.pingSynths[species.category];
  if (!synth) return;
```

```
    const pitches = this.pingPitches[species.category] || ['A4'];
    const pitch = pitches[Math.floor(Math.random() * pitches.length)];
    synth.triggerAttackRelease(pitch, '4n');
  }

  // Modulate master filter based on cursor depth
  modulateFilter(depth) {
    if (!this.masterFilter) return;
    const freq = 80 + (1 - depth) * 2200;
    this.masterFilter.frequency.rampTo(freq, 0.4);
  }

  dispose() {
    this.drone?.releaseAll();
    Object.values(this.zoneLayers).forEach(l => {
      try { l.synth?.dispose(); l.gain?.dispose(); } catch(e) {}
    });
  }
}
```

2B. Update useAudioEngine.js

```
js
```

```
import { useRef, useCallback } from 'react';
import AudioEngine from '../audio/AudioEngine';

export function useAudioEngine() {
  const engineRef = useRef(null);

  const start = useCallback(async () => {
    if (!engineRef.current) {
      engineRef.current = new AudioEngine();
    }
    await engineRef.current.start();
  }, []);

  const updateZones = useCallback((proximityMap) => {
    engineRef.current?.updateZones(proximityMap);
  }, []);

  const triggerDiamondPing = useCallback((species) => {
    engineRef.current?.triggerDiamondPing(species);
  }, []);

  const modulateFilter = useCallback((depth) => {
    engineRef.current?.modulateFilter(depth);
  }, []);

  return { start, updateZones, triggerDiamondPing, modulateFilter };
}
```

2C. Update App.jsx audio integration

Replace the handleSpeciesNear / speciesPing flow with a proximity map system:

jsx

```
// In App.jsx, replace species audio handling:

const { start, updateZones, triggerDiamondPing, modulateFilter } = useAudioEngine();

// handleProximityUpdate is called from SonicMap every frame
// proximityMap: { [speciesId]: { species: obj, proximity: 0-1 } }
const handleProximityUpdate = useCallback((proximityMap) => {
  if (!started) return;
  updateZones(proximityMap);

  // Find species for tooltip (highest proximity > 0.3)
  let best = null, bestProx = 0;
  for (const { species, proximity } of Object.values(proximityMap)) {
    if (proximity > bestProx && proximity > 0.3) {
      bestProx = proximity; best = species;
    }
  }
  setSpeciesPing(best);
}, [started, updateZones]);

const handleDiamondCenter = useCallback((species) => {
  triggerDiamondPing(species);
  // Sonar burst at cursor position
  setSonarBurst({
    active: true,
    x: cursorPosRef.current.x,
    y: cursorPosRef.current.y,
    color: species.color
  });
  setTimeout(() => setSonarBurst(b => ({...b, active: false})), 2000);
}, [triggerDiamondPing]);
```

2D. Update SonicMap.jsx — proximity map + diamond detection

In the nearest species detection loop (inside p.draw), replace the current logic:

```
js
```

```
// Build a full proximity map for ALL species every frame
const proximityMap = {};
let diamondCenterSpecies = null;

for (const sp of marineData.species) {
  const sx = sp.x * W + s.panX;
  const sy = sp.y * H + s.panY;
  const dx = s.mx - sx;
  const dy = s.my - sy;
  const dist = Math.sqrt(dx*dx + dy*dy);
  const threshold = sp.radius * W * 1.2;
  const proximity = Math.max(0, 1 - dist / threshold);

  if (proximity > 0) {
    proximityMap[sp.id] = { species: sp, proximity };
  }

  // Diamond center detection: within 18px of the center dot
  if (dist < 18) {
    diamondCenterSpecies = sp;
  }
}

// Pass to callbacks
onProximityUpdate(proximityMap);
if (diamondCenterSpecies) onDiamondCenter(diamondCenterSpecies);

// For tooltip: find highest proximity
const topEntry = Object.values(proximityMap).sort((a,b) => b.proximity - a.proximity)[0];
s.nearestSpecies = topEntry?.species ?? null;
s.speciesProximity = topEntry?.proximity ?? 0;
```

Update SonicMap props: add `onProximityUpdate` and `onDiamondCenter` callbacks. Remove `onSpeciesNear` (replaced by `onProximityUpdate`). Update App.jsx to pass these new props.

PART 3 — BIOLUMINESCENT TOPOLOGY GLOW ON SPECIES HOVER

When cursor enters a species zone (proximity > 0.2), add a soft colored radial glow to the topology lines *around that species location*.

This replaces the GlitchCard component (remove GlitchCard entirely).

3A. In drawContours() in SonicMap.jsx

After drawing all contour lines, add a bloom pass for each active species:

```
js
```

```
// After the main contour loop, before drawing species markers:

// For each species in proximity, add colored luminescence to nearby contour lines
for (const sp of marineData.species) {
  const sx = sp.x * width;
  const sy = sp.y * height;
  const prox = (sp === s?.nearestSpecies) ? s.speciesProximity : 0;
  // Also check if we have a proximity map reference to get exact value

  if (prox < 0.05) continue;

  const [r, g, b] = sp.color;
  const glowRadius = sp.radius * width * 1.5;

  // Draw the same contour lines a second time, clipped/faded by distance from species center
  // Implementation: iterate contour grid cells near the species,
  // redraw those segments with species color at low alpha

  for (let row = 0; row < ROWS; row++) {
    for (let col = 0; col < COLS; col++) {
      const cx = (col + 0.5) * cellW;
      const cy = (row + 0.5) * cellH;
      const distToSpecies = Math.sqrt((cx-sx)**2 + (cy-sy)**2);
      const cellProx = Math.max(0, 1 - distToSpecies / glowRadius);
      if (cellProx < 0.05) continue;

      const glowAlpha = cellProx * cellProx * prox * 60; // max ~60 alpha

      // Redraw the contour segments in this cell with species color
      // Use same marching squares logic but with species-colored stroke
      for (let level = 0; level < LEVELS; level++) {
        const iso = 0.12 + (level / LEVELS) * 0.72;
        const v00 = field[row][col];
        const v10 = field[row][col+1];
        const v01 = field[row+1][col];
        const v11 = field[row+1][col+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 x0 = col * cellW, y0 = row * cellH;
        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 lineMap = {
        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 = lineMap[idx];
      if (!segs) continue;

      pg.stroke(r, g, b, glowAlpha);
      pg.strokeWeight(0.8);
      for (const [a, b2] of segs) pg.line(a.x, a.y, b2.x, b2.y);
    }
  }
 }
}
```

Note: This inner loop may be slightly expensive. If frame rate drops below 30fps, add an optimization: only run the glow pass for species where proximity > 0.2, and limit to a bounding box of ±(glowRadius) around the species center.

3B. Pass the proximity map into drawContours

drawContours currently receives nearestSpecies and speciesProximity. Change signature to also accept `proximityMapRef` (a ref to the current frame's proximityMap) so it can render glow for ALL nearby species, not just the nearest.

Update the drawContours call and signature accordingly.

Also pass `field` as a parameter (currently it may be scoped differently — make sure drawContours has access to the current frame's field array).

PART 4 — TOOLTIP TO BOTTOM-RIGHT, REDESIGNED

Move the species tooltip from cursor-following to fixed bottom-right.

Update SpeciesTooltip.jsx

```
Position: fixed, bottom: 32px, right: 32px
```

Width: 240px
Transition: opacity 0.4s ease, transform 0.4s ease
When hidden: opacity 0, transform: translateY(8px)
When visible: opacity 1, transform: translateY(0)

Content (simplified, matching new data fields):

```
┌──────────────────────────────────────┐
│                  ● DECLINING │ ← status dot + text, DM Mono 8px, top-right
│                              │
│ Eelgrass                     │ ← Fraunces italic 30px weight 200
│ Zostera marina               │ ← DM Mono 10px rgba(0,180,220,0.45)
│                              │
│ ──────────────────────────── │ ← 1px rule opacity 0.1
│                              │
│ Intertidal to 6m             │ ← depthRange: DM Mono 9px, accent gold color
│                              │
│ Each acre sequesters         │ ← fact text: Fraunces italic 11px
│ carbon at 35× the rate       │   weight 200, rgba(140,210,240,0.65)
│ of a tropical rainforest.    │   max 3 lines
│                              │
│ ◈ PLANT                      │ ← category: DM Mono 8px 0.35 opacity
└──────────────────────────────┘
```

Remove the x,y position props entirely (no longer needed).
Remove the viewport clamping logic.
The component is always fixed bottom-right.

In App.jsx, remove cursorX/Y from the SpeciesTooltip render call.

PART 5 — INTRO SCREEN REFINEMENT

5A. Shorten the intro text in IntroScreen.jsx

Replace the body text with this shorter version:

```
The Puget Sound is an inland sea —
glacier-carved channels where Pacific
saltwater meets freshwater at haloclines.

Fourteen species. Each a node in a web
so entangled that losing one pulls
threads through all the others.
```

5B. Redesign the entry button

Remove the current button entirely.
Replace with a single minimal text link, positioned bottom-right of the left panel:

jsx

```jsx
<div style={{
  position: 'absolute',
  bottom: 52,
  right: 48,   // right edge of left panel
  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',
  userSelect: 'none',
}}
onMouseEnter={e => e.target.style.color = 'rgba(0, 210, 255, 0.9)'}
onMouseLeave={e => e.target.style.color = 'rgba(0, 180, 220, 0.5)'}
onClick={handleClick}
>
  Enter ↓
</div>
```

5C. Right panel decorative element — make it more refined

Replace the current static ellipses with an SVG animated topographic sketch:

jsx

```
// Right panel: full height, draws 5 nested organic oval paths
// using SVG with a slow draw-on animation (stroke-dashoffset)
// Colors: rgba(0,140,200,0.08) outermost → rgba(0,180,220,0.18) innermost
// Stroke width: 0.5px
// One slow rotation: animation: rotate 120s linear infinite on the SVG group
// Center: a single small dot (3px) rgba(0,180,220,0.4)
// The ovals are slightly different aspect ratios and rotation offsets
// giving an organic non-perfect topographic feel
```

5D. Remove bottom-left corner text from the main map view

In App.jsx / HUD.jsx: remove the `▲ SPECIES — LATIN NAME` text that appears bottom-left (this info now lives in the bottom-right tooltip).

5E. Remove the "?" help toggle button entirely

Delete the toggle button from IntroPanel.jsx.
The IntroPanel auto-hides and is not re-accessible.
(User can refresh to see it again — this is intentional for immersion.)

PART 6 — LARGER WORLD CANVAS (replacing true infinite canvas)

Instead of infinite tiling, implement a world canvas that is 2.5× the viewport,
with the Puget Sound content centered in it, and smooth drag-pan to explore.

6A. In SonicMap.jsx

Change canvas setup:

```
js
```

```
// World dimensions
const WORLD_W = window.innerWidth * 2.5;
const WORLD_H = window.innerHeight * 2.5;

p.setup = () => {
  // Canvas is still viewport size (for display)
  const cnv = p.createCanvas(window.innerWidth, window.innerHeight);
  cnv.parent(containerRef.current);

  // But the field resolution covers the larger world
  // Species positions are in WORLD space
  // Start panned so Puget Sound content is visible (centered-ish)
  s.panX = -(WORLD_W - window.innerWidth) / 2;
  s.panY = -(WORLD_H - window.innerHeight) / 2;

  ghostLayer = p.createGraphics(window.innerWidth, window.innerHeight);
  ghostLayer.background(0, 3, 13);
}
```

In buildField, use `WORLD_W` and `WORLD_H` instead of `W` and `H` for species influence calculations:

```
js

const mxWorld = s.mx - s.panX;  // mouse position in world space
const myWorld = s.my - s.panY;
const mxN = mxWorld / WORLD_W;
const myN = myWorld / WORLD_H;
```

Expand pan clamp to allow full world exploration:

```
js

s.panX = Math.max(-(WORLD_W - W), Math.min(0, s.panX));
s.panY = Math.max(-(WORLD_H - H), Math.min(0, s.panY));
```

Species positions: update all x,y in MarineData.json (already set in Part 1) — they map to WORLD space when multiplied by WORLD_W/WORLD_H.

The topology (Perlin noise) already tiles seamlessly so the ocean floor extends naturally beyond species data points into open water.

6B. Update species proximity detection to use world coordinates

```
js
```

```
for (const sp of marineData.species) {
  const spWorldX = sp.x * WORLD_W;
  const spWorldY = sp.y * WORLD_H;
  // Convert to screen space for distance check
  const spScreenX = spWorldX + s.panX;
  const spScreenY = spWorldY + s.panY;
  const dx = s.mx - spScreenX;
  const dy = s.my - spScreenY;
  ...
}
```

EXECUTION ORDER

1. Part 1 (data) — safe, JSON only
2. Part 2 (audio overhaul) — test audio thoroughly after this
3. Part 4 (tooltip position) — simple layout change
4. Part 5 (intro screen) — text + button only
5. Part 3 (bioluminescent glow) — test performance, may need optimization
6. Part 6 (larger canvas) — test pan/species detection carefully
7. Final `npm run build`

KNOWN LIKELY ISSUES

- Part 2: AudioEngine zone layers — if you get "cannot call triggerAttack on disposed synth" errors, add a try/catch around all synth calls and add an `isDisposed` flag check before calling.
- Part 3: If frame rate drops below 40fps after adding the glow pass, add this guard at the top of the glow loop: `if (p.frameRate() < 35) { skip glow for all but nearest species }`
- Part 6: Ghost layer trails will look odd when panning because the ghost is in screen space. Fix: clear ghostLayer more aggressively when `(Math.abs(s.panX - s.lastPanX) + Math.abs(s.panY - s.lastPanY)) > 3` Track lastPanX/Y in stateRef.

Report all judgment calls.