```
import { useCallback, useRef } from "react";
import { useSureSailStore } from "../store/sureSailStore";
import { dependencyMap } from "./dependencyMap";
import { playOceanOncePerRun } from "./oceanAudio";

type RunSelections = {
  domain: "SOL" | "BTC" | "ETH" | "MULTI";
  behavior: "WHALE" | "UNDERCHAIN" | "RETAIL";
};

function sleep(ms: number) {
  return new Promise((r) => setTimeout(r, ms));
}

export function useRunOrchestrator() {
  const pulseEdgeRef = useRef<((from: string, to: string) => void) | null>(null);

  const setPulseEdgeEmitter = useCallback((fn: (from: string, to: string) => void) => {
    pulseEdgeRef.current = fn;
  }, []);

  const startFoundationRun = useCallback(
    async (selections: RunSelections) => {
      const store = useSureSailStore.getState();

      // Disallow running in docs mode
      if (store.ui.dashboardMode === "docs") return;

      store.resetForNewRun();

      // PHASE 0: preflight + run id
      store.setPhase("PREFLIGHT");
      store.setPanelState("N_STATUS", "loading");

      const runRes = await fetch("/api/run", {
        method: "POST",
        body: JSON.stringify({ selections }),
        headers: { "Content-Type": "application/json" }
      });
      if (!runRes.ok) {
        store.setError("Failed to create run context", "N_STATUS");
        return;
      }
      const { run_id } = (await runRes.json()) as { run_id: string };
      store.setRunId(run_id);

      // PHASE 1: parallel 10CAT + GSS
      store.setPhase("PHASE1");
      store.setPanelState("N_10CAT", "executing");
      store.setPanelState("N_GSS", "executing");
      store.setPanelState("N_CTRL", "executing");

      // Ocean audio cue at PHASE 1 start if enabled
      const ui = useSureSailStore.getState().ui;
```

```
if (ui.neon && ui.audio === "OCEAN") {
  // Must be called from user click path; this is still within the click->async chain
  await playOceanOncePerRun(run_id, ui.volume);
}

// pulse dependencies from CTRL
pulseEdgeRef.current?.("N_CTRL", "N_10CAT");
pulseEdgeRef.current?.("N_CTRL", "N_GSS");

// Parallel calls
const [tenCatRes, gssRes] = await Promise.all([
  fetch("/api/10cat", {
    method: "POST",
    body: JSON.stringify({ run_id, selections }),
    headers: { "Content-Type": "application/json" }
  }),
  fetch("/api/gss", {
    method: "POST",
    body: JSON.stringify({ run_id, selections }),
    headers: { "Content-Type": "application/json" }
  })
]);

if (!tenCatRes.ok) {
  store.setPanelState("N_10CAT", "error");
  store.setError("10 CAT failed", "N_10CAT");
  return;
}
if (!gssRes.ok) {
  store.setPanelState("N_GSS", "error");
  store.setError("GSS failed", "N_GSS");
  return;
}

const tenCat = await tenCatRes.json();
const gss = await gssRes.json();
store.setTenCat(tenCat);
store.setGss(gss);

store.setPanelState("N_10CAT", "complete");
store.setPanelState("N_GSS", "complete");

// pulse into LABEL
pulseEdgeRef.current?.("N_10CAT", "N_LABEL");
pulseEdgeRef.current?.("N_GSS", "N_LABEL");

// PHASE 2: assemble + label + unlock
store.setPhase("PHASE2");
store.setPanelState("N_LABEL", "executing");

await sleep(120); // small stagger for "wave" effect

const asmRes = await fetch("/api/assemble", {
  method: "POST",
        body: JSON.stringify({ run_id, tenCat, gss }),
        headers: { "Content-Type": "application/json" }
      });
      if (!asmRes.ok) {
        store.setPanelState("N_LABEL", "error");
        store.setError("Assemble failed", "N_LABEL");
        return;
      }
      const foundation = await asmRes.json();
      store.setFoundation(foundation);

      store.setPanelState("N_LABEL", "complete");

      // pulse to progress
      pulseEdgeRef.current?.("N_LABEL", "N_FOUNDATION_PROGRESS");
      store.setPanelState("N_FOUNDATION_PROGRESS", "executing");
      await sleep(220);
      store.setPanelState("N_FOUNDATION_PROGRESS", "complete");

      // Unlock Stage II
      store.unlockStageII();

      // PHASE 4 finalize (for prototype we skip deep tools auto-run)
      store.setPhase("PHASE4");
      store.setPanelState("N_STATUS", "complete");
      store.setPanelState("N_CTRL", "complete");
      await sleep(250);
      store.setPanelState("N_CTRL", "idle");
      store.setPanelState("N_STATUS", "idle");
      store.setPhase("IDLE");
    },
    []
  );

  return { startFoundationRun, setPulseEdgeEmitter, dependencyMap };
}
```