CE2: Links, Hover, and Pursuit-like Navigation

Overview

With the core visualization components proven (treemap, beeswarm, circlepack-in-simulation, chord, adjacency) and the SceneCoordinator navigation streamlined, the next phase focuses on three pillars:

1. **Links** - Revealing and shaping relationships
2. **Hover** - Coordinated discovery across views
3. **Pursuit-like Navigation** - Search and documentation integration

Design principle: Minimal UI chrome. Navigation through direct manipulation of visualization elements, with search as the one exception for ad-hoc navigation.

Current State

Proven Reusable Components

- **Treemap** - Package-level (GalaxyTreemap, PkgTreemap)
- **Beeswarm** - Package-level with scope filtering (GalaxyBeeswarm)
- **Circlepack-in-simulation** - BubblePackBeeswarm (packages with modules inside)

Working but Not Yet Proven Reusable

- **Chord diagram** - Currently at package level, should work at module level
- **Adjacency matrix** - Currently at package level, should work at module level

Available Library Support

- Coordinated hover affordances in psd3-selection
- Force simulation with link support in psd3-simulation

Target State

Links

Two distinct types:

Discovery Links (Type 1)

- Ephemeral, shown on hover
- Reveal relationships without cluttering the view
- Useful when relationships are too numerous to show all at once

Force Links (Type 2)

- Persistent, part of the simulation
- Shape the layout
- Auto-enabled for sufficiently small neighborhoods (starting heuristic: neighborhood views only)

Future possibility: "Click to pin" to promote a discovery link to a force link. No current use case, but architecture shouldn't preclude it.

Hover

Graduated sophistication:

1. **Single-view highlighting** - Hover a package → highlight its direct deps in the same view
2. **Cross-view coordination** - Hover in chord → highlights in adjacency + bubblepack simultaneously
3. **Link materialization** - Hover reveals discovery links between related elements

Pursuit-like Navigation

Bidirectional navigation surface:

- **Viz → Panel**: Click module in treemap → slide-out panel shows source/documentation
- **Panel → Viz**: Click declaration in Pursuit result → viz navigates to that location

Components:

- Slide-out panel (source code view or Pursuit-style documentation)
- Search input (minimal, possibly hotkey-triggered)
- Local Pursuit integration (iframe with link interception)

Implementation Plan

Phase 1: Module-Level Enhancements

Goal: Prove chord/adjacency reusability, add hover highlighting, enable discovery links.

Tasks:

1. Apply chord diagram to single-package module view
2. Apply adjacency matrix to single-package module view
3. Add hover highlighting in module treemap (hover module → highlight its imports)
4. Add discovery links in neighborhood view (show import links on hover)

Key files:

- `SceneCoordinator.purs` - Wire up module-level chord/adjacency
- New or adapted chord/adjacency viz for module data
- Hover handlers in module treemap

Phase 2: Documentation Panel

Goal: Build the slide-out panel with source code view and Pursuit integration.

Tasks:

1. **Slide-out panel shell** - Halogen component with slide animation
2. **Source code view** - Click module in treemap → display source
    - Need to determine: source indexed at ingestion, or fetched on demand?

3. **Local Pursuit integration**
    - Run Pursuit locally serving documentation
    - Embed in iframe
    - Intercept internal link clicks
4. **Viz ← Panel navigation** - Pursuit clicks update the scene
    - Parse clicked link (package/module/declaration)
    - Map to scene navigation (find package, navigate to it, highlight module)

Key decisions:

- Source storage: Index at ingestion time vs fetch from GitHub/local filesystem
- Pursuit content: iframe embed (fast POC) vs parsed/restyled (better UX, more work)

Panel content (Pursuit-style):

- Module name header
- Package + repository links
- Type signatures with kinds
- Description (from doc comments)
- Member listings

Phase 3: Coordinated Hover

Goal: Wire up cross-view hover using library affordances.

Tasks:

1. **Cross-view hover wiring** - Chord + adjacency + bubblepack coordination
2. **Hover → temporary links** - Show relationship links on hover in force layouts
3. **Hover state propagation** - Ensure panel can react to hover (show quick info)

Library usage:

- psd3-selection coordinated hover affordances
- Hover event → broadcast to all views → each view highlights relevant elements

Phase 4: Search Integration

Goal: Complete the navigation loop with search.

Tasks:

1. **Search input** - Minimal chrome, hotkey-triggered (e.g., `/` or `Cmd+K`)
2. **Route to Pursuit** - Query → Pursuit search results in panel
3. **Result → Viz navigation** - Click search result → navigate viz to that location
4. **Type signature search** - Leverage Pursuit's powerful type search

UX flow:

1. User presses `/` → search input appears
2. Types query → results stream into panel
3. Clicks result → panel shows full docs, viz navigates to package/module

4. Clicks declaration in docs → viz highlights that location

Architecture Notes

Panel as Navigation Surface

The slide-out panel is not just a detail view - it's a bidirectional navigation surface:

```
┌─────────────────────────────────────────┐
│                                         │
│         SVG Visualization               │
│                                         │
│    click module ──────────────────┐     │
│                                   │     │
│    ◄─── navigate to location ─────┼──┐  │
│                                   │  │  │
└───────────────────────────────────┼──┼──┘
                                    │  │
┌───────────────────────────────────▼──┼──┐
│  Slide-out Panel                     │  │
│  ┌────────────────────────────────┐  │  │
│  │ Module: Data.Foldable          │  │  │
│  │ Package: foldable-traversable  │  │  │
│  │                                │  │  │
│  │ class Foldable f where         │  │  │
│  │   foldr :: ...                 │──┘  │
│  │   foldl :: ...    ◄── click       │  │
│  └────────────────────────────────┘     │
└─────────────────────────────────────────┘
```

Link Interception Strategy

For Pursuit iframe integration:

```
// In panel component
iframe.contentWindow.document.addEventListener('click', (e) => {
  const link = e.target.closest('a');
  if (link && isInternalPursuitLink(link.href)) {
    e.preventDefault();
    const {package, module, declaration} = parsePursuitUrl(link.href);
    // Route to our navigation
    navigateToLocation(package, module, declaration);
  }
});
```

Small Neighborhood Heuristic

Starting simple: enable force links only in `PkgNeighborhood` scenes. This gives a natural boundary (focal package + direct deps + direct dependents) that's likely to be manageable.

Future refinement: count-based or density-based thresholds.

Dependencies

- **Library fixes** (in progress): Array safety fixes in psd3-simulation/psd3-selection
- **Local Pursuit**: Need to set up and run locally for development
- **Source indexing**: Decision needed on whether to index source at ingestion

Future Scope (Not This Plan)

- Click-to-pin links (no current use case)
- Ingestion UI (separate app/tab)
- Semantic zoom to actual code (ZUI paradigm)
- Reverse navigation animations

Success Criteria

1. Chord/adjacency work at module level in single-package view
2. Hover in any view highlights related elements
3. Click module → see source/docs in panel
4. Search → results → click → viz navigates
5. Feels like direct manipulation, minimal chrome

Status

Status: Ready to implement after library fixes complete.

Next action: Begin Phase 1 (module-level enhancements) once psd3-simulation/psd3-selection work is merged.