Linear Graph Considerations

This note summarizes how to think about (and implement) linearized pangenome views using the `PangenomeService`. It's meant for engineers wiring up analysis → rendering in a three.js app.

---

1) The "spine" is any assembly you choose

- The **spine** is simply the assembly walk you pass into `assessGraphFeatures(spineWalk, ...)`.
- All base-pair (bp) positions in the output (`spine.nodes[].bpStart/bpEnd`, `events[*].anchors.spanStart/spanEnd`) are measured along **that** spine's axis.
- Swapping GRCh38 for a sample assembly (e.g., `HG01106#1#JAHAMC…`) re-anchors everything without code changes.

Tip (multi-component assemblies):
Some assemblies split into multiple connected components. Either:

- pick the **main** path (usually longest) and pass `[mainPath]`, or
- run `assessGraphFeatures` per path and render per-component rows/lanes.

---

2) Coordinate systems & spans

- For a spine node `S`:
  `bpStart(S)` and `bpEnd(S)` accumulate true bp lengths of preceding spine nodes.
- For an event with anchors `(L,R)` on the spine:
  `spanStart = bpEnd(L)`, `spanEnd = bpStart(R)`, `refLenBp = max(0, spanEnd − spanStart)`.
- A **pill** has `refLenBp = 0` (adjacent anchors); it's an insertion relative to the spine.

Path projection (for rendering):
Each sampled alternative path (`event.paths[*]`) includes `nodesDetailed[]` with:

- `refBpStart/refBpEnd`: projection to `[spanStart, spanEnd]` (bp on the spine).
- `altStartBp/altEndBp`: cumulative distance along the path itself (useful for width/labels).

---

3) What "multiple paths" and "disjoint" really mean

- **Assembly walks** (`createAssemblyWalk`) return **one path per connected component** in that assembly's induced subgraph.
  Paths are always **connected** (no "gappy" walks). Singletons are allowed (`["node"]`).
- Within an **event**, the projection of alt-path interior nodes is **non-overlapping per path** by construction. Different events can overlap or nest (that's what we call **braids**).
- With `allowMidSpineReentry: true`, a path can include intermediate **spine** nodes; their true spine extents may overlap the off-spine projections visually—this is expected and often desirable to show complex structure.

---

4) Event taxonomy (UI-friendly)

- **pill**: adjacent anchors; zero-length on the spine; one or more non-zero-length alt paths (insertions).
- **simple_bubble**: a single detour between `(L,R)`; disjoint from other events.
- **parallel_bundle**: multiple paths share the same `(L,R)` (parallel alleles).
- **braid**: event span **overlaps** or **nests** another event span (interleaving or containment).
- **dangling**: branch that leaves the spine but doesn't rejoin inside the current window.

> Classification is based on **interval relations in spine bp space**, not only local shape.

---

5) Pills: zero-bp anchors with real sequence

- `bpStart == bpEnd` at the junction; insertion length lives in `paths[*].altLenBp`.
- For a single-node insertion, `altLenBp` equals that node's length.
- When rendering, position the pill at `spanStart` and size/height by `altLenBp` (or by number of parallel paths).

---

6) Mirror pairs & duplicates

With `includeUpstream: true`, you may detect both `(L→R)` and `(R→L)` for the same junction (especially pills). Either:

- keep both (and display directions), or
- **dedupe** to one canonical event and record that both orientations were seen.

---

7) Windowing effects & completeness

- The analyzer works on your current **graph window**.
  - Entries that **exit** the window become **dangling**.
  - Off-spine components that **never touch** the spine are reported in `offSpine[]` (context).
- You can raise `maxRegionNodes/Edges` and `maxPathsPerEvent` for deeper exploration. Truncations set `event.region.truncated` or `event.stats.truncatedPaths`.

---

8) Rendering playbook (three.js)

X-axis mapping

```
const x = (bp) => (bp - locusStartBp) * pxPerBp;
```

Spine

- Draw pink bars for each `spine.nodes[i]` from `x(bpStart)` to `x(bpEnd)`.
- Optionally annotate node id/length.

Events

- For `(L,R)` events: draw arcs (or ribbons) from `x(spanStart)` to `x(spanEnd)`.
- **Lane packing:** assign a lane per event via greedy interval packing on `[spanStart, spanEnd]`.
- Color by `type` (pill, bubble, braid) or by `Δlen = minAltLenBp − refLenBp`.

Alt paths

- For an expanded view, lay interior segments using `nodesDetailed[].refBpStart/refBpEnd`.
- For pills, all segments project to the same x; visualize as stacked beads or a "balloon" with width from `altLenBp`.

Off-spine context

- Render `region.nodes` faintly when an event is selected.
- Use `region.anchorEdges` to draw connectors back to anchors/mid-spine.

---

9) Options that impact the look/feel

- `allowMidSpineReentry`
  - **true** → more realistic complexity; more events become **braids**.
  - **false** → cleaner, "pure detours" view.
- `includeAdjacent`
  - **true** → shows pills (insertions) explicitly.
  - **false** → hides junction-level insertions.
- `includeUpstream`
  - **true** → both directions; dedupe if you don't want pairs.
- `includeDangling`
  - **true** → users see where structure continues off-window.

Performance knobs

- `maxPathsPerEvent` (3–5 for UI; up to 8+ for analysis).
- `maxRegionNodes/Edges` (raise when inspecting huge superbubbles).

---

10) Data quality & assumptions

- Node length = `node.length` or `sequence[id].length` (fallback 0).
  If spine nodes have length 0, their bp extents collapse. Ensure lengths are present for spine candidates.
- Node ids include orientation (`+/−`). If you reconstruct sequences for alt paths, reverse-complement interior nodes with `−`.
- Paths are **connected** (no gaps). Singletons are emitted for isolated keyed nodes.

---

11) Switching spines (UX)

- Label the axis clearly: “Spine: `<assembly key>`”.
- Reset cached `features` on spine change; re-run `assessGraphFeatures`.
- If an assembly has multiple components, render them as separate rows or let the user pick the “main” path.

---

12) Quick end-to-end

```
const svc = new PangenomeService(json);
const walk = svc.createAssemblyWalk(userChosenKey, { mode: "auto" });

// Choose a path:
const main = walk.paths.reduce((a,b)=> (a?.bpLen||0) > b.bpLen ? a : b, null);
const spineWalk = { ...walk, paths: main ? [main] : [] };

const features = svc.assessGraphFeatures(spineWalk, {
  includeAdjacent: true,
  includeUpstream: true,          // dedupe later if desired
  allowMidSpineReentry: true,
  includeDangling: true,
  includeOffSpineComponents: true,
  maxPathsPerEvent: 5
});

// Render spine + events using bp→px mapping and lane packing.
```

---

13) Sanity checklist

- [ ] Spine nodes have non-zero lengths (or sequences) → credible bp axis.
- [ ] Decide policy for multi-component spines (pick longest vs per-component view).
- [ ] Decide `allowMidSpineReentry` (clarity vs completeness).
- [ ] Dedupe mirror pills if `includeUpstream: true`.
- [ ] Surface truncation flags in the UI (`⚠ truncated`).
- [ ] For pills, use `paths[*].altLenBp` to size visuals.
- [ ] Keep interaction snappy: start with `maxPathsPerEvent: 3–5`.

If you want, I can produce a companion “Renderer Recipes” doc with concrete three.js snippets for arcs, interval packing, and pill glyphs.