07 Algorithmen und Layout-Engine

Die Art und Weise, wie Knoten angeordnet sind, bestimmt massgeblich die Lesbarkeit eines Graphen. Nodges bietet eine leistungsfähige Layout-Engine, die verschiedene Algorithmen unterstützt und rechenintensive Aufgaben effizient parallelisiert.

07.1 Layout-Manager Architektur

Der `LayoutManager` ist für die Berechnung der Positionen (x, y, z) aller Knoten verantwortlich. Er folgt dem **Strategy Pattern**: Unterschiedliche Algorithmen (Strategien) können zur Laufzeit ausgetauscht werden, ohne dass der Rest der Anwendung angepasst werden muss.

Synchron vs. Asynchron

- **Synchrone Algorithmen**: Einfache geometrische Anordnungen (wie Grid oder Circle) werden direkt im Haupt-Thread berechnet, da sie mathematisch trivial und extrem schnell sind (< 10ms).
- **Asynchrone Algorithmen**: Komplexe Simulationen (wie Force-Directed Layouts), die hunderte Iterationen benötigen, laufen asynchron im Hintergrund, um das UI nicht einzufrieren.

07.2 Web Worker Integration

Physik-Simulationen sind teuer ($O(n^2)$ oder $O(n \log n)$). Wenn sie im Haupt-Thread (Main Thread) laufen würden, würde die Benutzeroberfläche und das Rendering ruckeln oder komplett blockieren.

Multithreading

Nodges lagert diese Berechnungen in **Web Workers** aus. Ein Web Worker ist ein Skript, das in einem separaten Thread läuft.

1. **Bootstrapping**: Beim Start initialisiert der LayoutManager einen Worker.
2. **Daten-Transfer**: Die Knoten- und Kanten-Daten werden an den Worker gesendet (mittels `postMessage`).
3. **Iteration**: Der Worker berechnet einen Simulations-Schritt (Tick).
4. **Synchronisation**: Nach jedem Schritt sendet der Worker die neuen Positionen zurück an den Main Thread.
5. **Rendering**: Der `NodeObjectsManager` aktualisiert die 3D-Szene live. Der Benutzer sieht, wie sich der Graph "entfaltet".

07.3 Implementierte Algorithmen

Nodges stellt eine breite Palette an Algorithmen bereit, um unterschiedliche Datenstrukturen optimal darzustellen.

Force-Directed Layouts

Diese Algorithmen simulieren physikalische Kräfte: Knoten stossen sich ab (wie Magnete), Kanten ziehen verbundene Knoten zusammen (wie Federn).

- **ForceAtlas2 (ähnlich)**: Eignet sich hervorragend für Cluster-Bildung.
- **Fruchterman-Reingold**: Ein Klassiker, der ästhetisch ansprechende, symmetrische Graphen erzeugt.
- **Spring-Embedder**: Einfaches Feder-Masse-Modell.

Vorteil: Deckt organische Strukturen und Cluster automatisch auf.

Strukturelle Layouts

Diese Algorithmen ordnen Knoten deterministisch nach festen Regeln an.

- **Grid (Raster)**: Ordnet Knoten in einem 3D-Würfelraster an. Ideal, um Mengenverhältnisse abzuschätzen.
- **Circular (Kreis)**: Alle Knoten auf einem Ring. Gut für kleine Netzwerke.
- **Spherical (Kugel)**: Verteilt Knoten auf der Oberfläche einer Kugel (Fibonacci Sphere).
- **Helix**: Anordnung in einer Spirale (z.B. für Zeitreihen).

Hierarchische Layouts (Tree)

- Für Daten mit klarer Eltern-Kind-Beziehung (Bäume).
- **3D-Tree**: Nutzt die Z-Achse für die Tiefe oder ordnet Ebenen radial an (Cone Tree).

Ende Kapitel 07