

Interaktions-Guide für Nodges

Überblick

Dieses Dokument erklärt, wie die wichtigsten Interaktionen im **Nodges**-Projekt implementiert sind und wie Sie sie in Ihrem Code nutzen können:

- **Hover** (Schweben über Knoten/Edges)
- **Selektieren** (Klick-Auswahl)
- **Path-Suche** (Pfad zwischen Knoten finden)
- **Search** (Suche nach Knoten/Edges nach Namen oder Attributen)

Alle genannten Funktionen befinden sich im `src/core`-Verzeichnis und werden über den **StateManager** und die zugehörigen Manager-Klassen gesteuert.

1. Hover-Mechanik

1.1. Kern-Komponente

- **Datei:** `src/core/RaycastManager.ts`
- **Methode:** `findIntersectedObject(raycaster: THREE.Raycaster)`
 - Durchläuft die Szene mit einem `THREE.Raycaster` und liefert das erste intersected Objekt (Knoten oder Edge).
 - Unterscheidet zwischen `THREE.Mesh` (kurvige Edges) und `THREE.InstancedMesh` (geradlinige Edges).

1.2. Weiterleitung an HighlightManager

- **Datei:** `src/core/HighlightManager.ts`
- **Methode:** `onHover(object: THREE.Object3D)`
 - Prüft, ob das Objekt bereits hervorgehoben ist.
 - Erstellt ein temporäres Highlight-Objekt (z. B. `THREE.Line` für Edges oder `THREE.Mesh` für Nodes) mit einer leicht transparenten Farbe (`hoverColor`).
 - Für kurvige Edges wird die originale `curve`-Geometrie aus `object.userData.curve` verwendet, sodass das Highlight exakt der Kurve folgt.

1.3. Styling des Hover-Effekts

```
const hoverColor = new THREE.Color(0x4a90e2); // sanftes Blau
highlightMaterial = new THREE.MeshBasicMaterial({
  color: hoverColor,
  opacity: 0.4,
  transparent: true,
});
```

Der Farbwert kann in `src/config/theme.ts` angepasst werden.

2. Selektieren (Klick-Auswahl)

2.1. Event-Handling

- **Datei:** `src/core/CentralEventManager.ts`
- **Methode:** `onClick(event: MouseEvent)`
 - Nutzt denselben `Raycaster` wie beim Hover, ruft aber `HighlightManager.select(object)` auf.

2.2. Auswahl-Logik

- **Datei:** `src/core/HighlightManager.ts`
- **Methode:** `select(object: THREE.Object3D)`
 - Entfernt vorherige Auswahl-Highlights.
 - Setzt `object.userData.isSelected = true`.
 - Erstellt ein beständiges Highlight-Mesh mit einer anderen Farbe (z.B. `0xffffd700` – Gold).

2.3. Persistente Auswahl-Darstellung

```
const selectColor = new THREE.Color(0xffffd700);
const selectMaterial = new THREE.MeshBasicMaterial({
  color: selectColor,
  opacity: 0.6,
  transparent: true,
});
```

3. Pfad-Suche (Shortest Path)

3.1. API-Funktion

- **Datei:** `src/core/PathFinder.ts`
- **Klasse:** `PathFinder`
- **Methode:** `findShortestPath(startId: string, endId: string): string[]`
 - Implementiert den Dijkstra-Algorithmus auf dem internen Graph-Modell (`EntityData/RelationshipData`).
 - Gibt ein Array von Knoten-IDs zurück, das den kürzesten Pfad darstellt.

3.2. Visualisierung des Pfads

- **Datei:** `src/core/PathHighlighter.ts`
- **Methode:** `highlightPath(nodeIds: string[])`
 - Durchläuft die Knoten-IDs, ruft für jede zugehörige Edge das Highlight-Material mit einer **Pfad-Farbe** (`0x00ffff` – Grün) auf.
 - Entfernt alte Pfad-Highlights, bevor ein neuer Pfad angezeigt wird.

4. Allgemeine Suche (Search)

4.1. Such-Service

- **Datei:** `src/core/SearchService.ts`
- **Klasse:** `SearchService`
- **Methoden:**
 - `searchByName(query: string): EntityData[]` – filtert Knoten/Edges nach Namen.
 - `searchByAttribute(attr: string, value: any): EntityData[]` – filtert nach beliebigen Attributen.

4.2. Nutzung im UI

- **Datei:** `src/ui/SearchPanel.tsx` (React-Komponente, falls das UI-Framework React verwendet)
- Bindet `onChange` an `SearchService` und gibt die Trefferliste an `HighlightManager` weiter, das die gefundenen Objekte temporär hervorhebt.

5. Zusammenfassung der wichtigsten Aufrufe

Aktion	Einstiegspunkt	Weiterleitung	Ergebnis
Hover	<code>RaycastManager.findIntersectedObject</code>	<code>HighlightManager.onHover</code>	Temporäres, leicht transparentes Highlight
Selektieren	<code>CentralEventManager.onClick</code>	<code>HighlightManager.select</code>	Permanentes Highlight mit anderer Farbe
Pfad-Suche	<code>PathFinder.findShortestPath</code>	<code>PathHighlighter.highlightPath</code>	Grün hervorgehobene Edges entlang des Pfads
Search	<code>SearchService.search*</code>	<code>HighlightManager.highlightMultiple</code> (intern)	Temporäres Highlight aller Treffer

6. Weiterführende Hinweise

- **Performance:** Hover- und Click-Events laufen im Render-Loop; stellen Sie sicher, dass das Highlight-Material nicht jedes Frame neu erstellt wird – verwenden Sie ein Pool-Pattern (`HighlightPool`).
- **Erweiterbarkeit:** Neue Interaktions-Typen (z. B. **Box-Select**) können durch Ergänzen von Methoden in `CentralEventManager` und `HighlightManager` implementiert werden.
- **Testing:** Unit-Tests für `PathFinder` und `SearchService` liegen in `src/tests/` und können mit `npm test` ausgeführt werden.

Dieses Dokument wurde am 2025-12-31 erstellt und soll als Referenz für Entwickler dienen, die mit den Interaktions-Mechanismen von Nodges arbeiten.