06 Interaktions-Design und Input-Processing

Eine nahtlose Interaktion ist für User Experience in 3D-Anwendungen entscheidend. Dieses Kapitel beschreibt, wie Nodges Benutzereingaben verarbeitet und in Aktionen innerhalb der 3D-Welt übersetzt.

06.1 Raycasting und Picking

Das größte technische Problem bei Interaktionen in 3D ist das "Picking": Wie stellt man fest, welches 3D-Objekt sich unter dem Mauszeiger befindet, der sich nur auf einem 2D-Bildschirm bewegt? Nodges nutzt hierfür **Raycasting**.

Das Prinzip

1. **Mauskoordinaten**: Die 2D-Pixel-Koordinaten (x, y) der Maus werden auf den normalisierten Gerätekoordinatenraum (NDC) von -1 bis +1 umgerechnet.
2. **Strahl aussenden**: Von der Kameraposition wird ein unsichtbarer Strahl (Ray) durch diese Koordinate in die 3D-Szene "geschossen".
3. **Schnittmenge**: Three.js prüft mathematisch, welche Objekte dieser Strahl durchschneidet.
4. **Sortierung**: Die getroffenen Objekte werden nach Entfernung zur Kamera sortiert. Das erste Objekt ist das "Gesehene".

Optimierungen

Da Raycasting rechenintensiv ist (Prüfung gegen tausende Meshes), wendet Nodges Optimierungen an:

- **Bounding Spheres**: Zuerst wird geprüft, ob der Strahl überhaupt die grobe Umhüllung eines Objekts trifft, bevor die exakte Geometrie geprüft wird.
- **Layering**: Raycasting wird nur auf relevante Layer (Knoten, Kanten) angewendet, Dekorations-Elemente werden ignoriert.
- **Schwellenwerte (Thresholds)**: Bei dünnen Linien (Lines) wird ein Toleranzbereich (Threshold) definiert, damit der Benutzer nicht pixelgenau klicken muss.

06.2 Input-Handler (`CentralEventManager`)

Der `CentralEventManager` fängt alle nativen Browser-Events ab und leitet sie in die Logik der Anwendung weiter.

Maus-Interaktionen

- **MouseMove**: Löst kontinuierlich Raycasting aus, um Hover-Zustände zu aktualisieren. Wird "throttled" (gedrosselt, z.B. alle 100ms), um die CPU-Last zu begrenzen.
- **Click**: Differenziert zwischen einem echten "Klick" und dem Ende eines "Drag"-Vorgangs. Wenn sich die Maus zwischen `mousedown` und `mouseup` signifikant bewegt hat, wird *kein* Klick-Event gefeuert (da der User vermutlich die Kamera drehen wollte).
- **ContextMenu**: Rechtsklicks können kontextsensitive Menüs öffnen (aktuell reserviert für zukünftige Features).
- **Wheel**: Steuert den Zoom der Kamera.

Keyboard-Shortcuts

Zur Power-User-Steuerung werden Tastatureingaben überwacht:

- `Leertaste`: Pausiert/Startet die Layout-Simulation.
- `R`: Setzt die Kamera zurück (Reset).
- `H`: Blendet UI-Elemente ein/aus.
- `Esc`: Hebt Selektionen auf oder schließt Panels.

06.3 Kamera-Steuerung

Die Kamera ist das Auge des Benutzers. Eine schlechte Kamerasteuerung führt sofort zu Orientierungsverlust ("Motion Sickness").

OrbitControls Integration

Nodges verwendet standardisierte `OrbitControls`. Diese erlauben eine intuitive Steuerung ähnlich wie in CAD-Software:

- **Orbit (Linke Maus + Drag)**: Drehung um den Mittelpunkt.
- **Pan (Rechte Maus + Drag)**: Verschieben der Kameraebene.
- **Zoom (Mausrad)**: Annäherung an den Fokuspunkt.

Automatische Kamera-Fahrten (Auto-Focus)

Wenn ein Benutzer einen Knoten markiert (z.B. über die Suche), führt die Kamera oft eine sanfte Animation durch:

1. **Ziel-Ermittlung**: Berechnung der Position des Zielknotens.
2. **Interpolation**: Die Kamera "fliegt" über einen Zeitraum von z.B. 1000ms von der aktuellen Position zur neuen, wobei Position und Blickrichtung (Target) interpoliert werden (mithilfe der *Tween.js* Bibliothek).
3. **Easing**: Die Bewegung startet langsam, beschleunigt und bremst am Ende sanft ab (Quadratic Ease-In-Out), um abrupte Bewegungen zu vermeiden.

Ende Kapitel 06