

---

# 1 Szenengraphen für interaktive 3D-Anwendungen am Beispiel X3DOM

---

## 1.1 Einführung und Motivation

---

Die 3D-Computergrafik ist ein integraler Bestandteil moderner Anwendungen, von Spielen bis hin zu industriellen Visualisierungen. Bei der Entwicklung solcher Systeme stehen Entwickler vor mehreren technischen Herausforderungen:

- **Effizienz:** Handhabung großer Datenvolumina und Echtzeit-Rendering.
- **Portabilität:** Lauffähigkeit auf Desktop-PCs, Tablets und Mobilgeräten.
- **Integration:** Nahtloses Einbetten von 3D-Inhalten in bestehende Anwendungen (z.B. Webseiten).

Ein zentraler Ansatz zur Bewältigung dieser Komplexität ist die Strukturierung der Daten mittels **Szenengraphen**.

## 1.2 Strukturierung von 3D-Szenendaten

---

Um eine 3D-Szene rendern zu können, benötigt die Grafikpipeline vielfältige Informationen. Diese lassen sich nicht als unstrukturierte Liste verwalten, da komplexe Beziehungen zwischen den Elementen bestehen.

### 1.2.1 Notwendige Informationen für das Rendering

---

- **Objekt-Geometrie:** Form der Objekte (z.B. Vertices, Polygone).
- **Transformationen:** Position, Rotation und Skalierung im Raum.
- **Materialien:** Oberflächenbeschaffenheit, Farben, Texturen.
- **Kameras:** Blickwinkel und Projektionsart.
- **Lichter:** Lichtquellen (Punktlicht, Richtung, Spot) und deren Eigenschaften.
- **Spezialeffekte:** Nebel, Schatten, Skyboxes.

### 1.2.2 Beziehungen zwischen Daten

---

In einer Szene treten häufig Redundanzen und Hierarchien auf:

- Mehrere Objekte teilen sich dasselbe Material (z.B. alle Reifen eines Autos sind schwarz).
- Dasselbe geometrische Objekt wird mehrfach instanziiert (z.B. vier identische Räder an verschiedenen Positionen).
- Objekte sind logisch gruppiert und bewegen sich gemeinsam (z.B. bewegt sich das Rad relativ zum Auto, das Auto relativ zur Welt).

## 1.3 Das Konzept des Szenengraphen

---

Ein Szenengraph ist die fundamentale Datenstruktur zur Organisation dieser Informationen.

## Szenengraph (Scene Graph)

Ein Szenengraph ist ein **gerichteter, azyklischer Graph** (Directed Acyclic Graph, DAG), der die logische und räumliche Struktur einer 3D-Szene beschreibt.

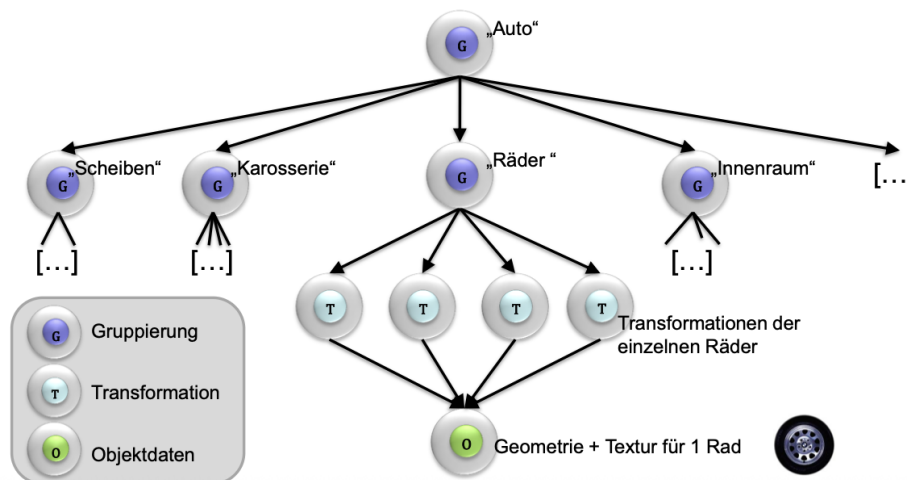
- **Gerichtet:** Kanten haben eine klare Richtung (Eltern → Kind).
- **Azyklisch:** Es gibt keine geschlossenen Wege (Schleifen); eine Endlosschleife beim Rendern wird verhindert.
- **Wurzelknoten:** Der Einstiegspunkt für die Traversierung.

Im Gegensatz zu einem reinen Baum kann ein Knoten im DAG (außer der Wurzel) **mehrere Elternknoten** besitzen. Dies ermöglicht die Wiederverwendung von Ressourcen (Instanziierung).

### 1.3.1 Knotentypen

Ein Szenengraph besteht üblicherweise aus drei abstrakten Kategorien von Knoten:

1. **G (Gruppierung):** Organisiert Unterobjekte logisch (z.B. "Auto"). Kann Bedingungen prüfen (z.B. Sichtbarkeit).
2. **T (Transformation):** Enthält geometrische Transformationen (Translation, Rotation, Skalierung).
3. **O (Objekt/Shape):** Die eigentlichen Blattknoten, die Geometrie und Material definieren (das, was gezeichnet wird).



### 1.3.2 Traversierung und Rendering

Der Prozess des Zeichnens wird als **Traversierung** bezeichnet. Dabei wird der Graph rekursiv durchlaufen, beginnend bei der Wurzel. **Ablauf der Traversierung:**

- Die Anwendung startet an der Wurzel.
- Jeder Kindknoten wird rekursiv besucht (Tiefensuche).
- Während des Abstiegs wird der globale Zustand aktualisiert, insbesondere die **Current Transformation Matrix (CTM)**.

**Operationen während der Traversierung:**

- **Bei Gruppierungsknoten:** Prüfe, ob die Gruppe aktiv ist. Falls ja, traversiere alle Kinder.
- **Bei Transformationsknoten:** Multipliziere die aktuelle CTM mit der lokalen Transformationsmatrix  $M$  des Knotens:

$$CTM_{neu} = CTM_{alt} \cdot M$$

Diese akkumulierte Matrix wird an die Kinder weitergegeben.

- **Bei Objektknoten:** Zeichne das Objekt unter Verwendung der aktuellen CTM. Dadurch wird das Objekt an der korrekten, akkumulierten Weltposition dargestellt.

### 1.3.3 Vorteile

- **Wiederverwendbarkeit:** Geometrien (z.B. ein Rad) müssen nur einmal im Speicher liegen und können mehrfach referenziert werden.
- **Semantische Gruppierung:** Einfaches Ein-/Ausblenden ganzer Baugruppen.
- **Transformationshierarchie:** Bewegt man einen Elternknoten (z.B. "Auto"), bewegen sich alle Kindknoten (z.B. "Räder") automatisch mit.

## 1.4 X3DOM: Szenengraphen im Web

X3DOM ist eine Technologie, die 3D-Szenengraphen direkt in HTML integriert.

### X3DOM

X3DOM ist eine Open-Source-Lösung, die den **X3D**-Standard direkt im HTML-DOM verfügbar macht. Es benötigt keine Plugins, sondern nutzt WebGL via JavaScript ("Polyfill"-Layer), um 3D-Inhalte nativ im Browser zu rendern.

### 1.4.1 Hintergrund: VRML und X3D

- **VRML (Virtual Reality Modeling Language, 1994/97):** Erster Standard für 3D im Web.
- **X3D (Extensible 3D):** Der XML-basierte Nachfolger von VRML. Er ist modular (Profile) und unterstützt verschiedene Encodings (XML, Binary, VRML-Classic).

### 1.4.2 Integration in HTML

X3DOM ermöglicht eine **deklarative** Beschreibung der Szene. Das bedeutet, man beschreibt *was* dargestellt werden soll (Struktur), nicht *wie* es gezeichnet wird (Imperative Befehle wie in OpenGL/WebGL direkt).

Grundstruktur eines X3DOM-Dokuments:

```
<!DOCTYPE html><html><head><script src='`x3dom.js`'></script><link rel='`stylesheet`'  
→ href='`x3dom.css`'></head><body><x3d width='400px'  
→ height='400px'><scene><shape><appearance><material diffuseColor='1 0  
→ 0'></material></appearance><box></box></shape></scene></x3d></body></html>
```

## 1.5 Wichtige X3D-Knoten

Der Aufbau einer Szene in X3DOM folgt einer strikten Hierarchie:

### 1.5.1 Shape (Gestalt)

Der `<Shape>`-Knoten verbindet Geometrie mit Aussehen. Er enthält in der Regel zwei Kindknoten:

1. **Geometry:** Die Form (z.B. `<Box>`, `<Sphere>`, `<Cone>`, `<IndexedTriangleSet>`).
2. **Appearance:** Das Aussehen.

### 1.5.2 Appearance und Material

Der `<Appearance>`-Knoten beinhaltet den `<Material>`-Knoten, der die optischen Eigenschaften nach dem **Phong-Beleuchtungsmodell** definiert. Das Phong-Modell setzt sich zusammen aus:

$$I_{out} = I_{ambient} + I_{diffuse} + I_{specular}$$

Parameter im `<Material>`-Tag:

- **diffuseColor:** Die eigentliche Farbe des Objekts unter direkter Beleuchtung.
- **specularColor:** Farbe des Glanzlichts (meist weiß).
- **shininess:** Größe des Glanzpunkts (Rauheit der Oberfläche).
- **emissiveColor:** Selbstleuchten (unabhängig von Lichtquellen).
- **transparency:** Durchsichtigkeit.

### 1.5.3 Komplexe Geometrie: IndexedTriangleSet

---

Für beliebige Formen, die nicht durch Primitive (Box, Kugel) darstellbar sind, werden Polygonnetze verwendet.

- **Coordinate Point:** Liste aller Eckpunkte (Vertices) im 3D-Raum (x y z).
- **Index:** Definiert, welche Punkte zu einem Dreieck verbunden werden. Dies spart Speicher, da Punkte wiederverwendet werden können.
- **Normal:** Normalenvektoren für die Lichtberechnung.

*Hinweis:* Große Geometrien sollten aus Performancegründen in externe binäre Dateien ausgelagert werden (ähnlich wie Bilder via `src`).

### 1.5.4 Transformationen

---

Der `<Transform>`-Knoten manipuliert das Koordinatensystem für seine Kinder.

- **translation='x y z':** Verschiebung.
- **rotation='x y z angle':** Rotation um Achse (x,y,z) mit Winkel (Bogenmaß).
- **scale='x y z':** Skalierung.

Durch Verschachtelung von Transform-Knoten entstehen kinematische Ketten (z.B. Oberarm → Unterarm → Hand).

## 1.6 Instanziierung im DOM (DEF/USE)

---

Ein fundamentales Problem bei der Abbildung von Szenengraphen auf das HTML-DOM ist die Struktur:

- **HTML-DOM:** Ist ein Baum. Jedes Element hat genau *einen* Elternknoten.
- **Szenengraph:** Ist ein DAG. Ein Geometrie-Knoten kann von mehreren Transform-Knoten referenziert werden (Instanziierung).

### 1.6.1 Lösung: DEF und USE Mechanismus

---

X3D löst dies durch Referenzierung:

1. **DEF (Definition):** Ein Knoten wird definiert und benannt. Er wird an dieser Stelle normal verarbeitet. `<Group DEF='AutoRad'> ...</Group>`
2. **USE (Verwendung):** An einer anderen Stelle im Baum wird eine Instanz eingefügt, die auf die Definition verweist. `<Group USE='AutoRad'></Group>`

Dies ermöglicht es, komplexe Objekte (z.B. "Suzanne mit Hut") einmal zu definieren (Gruppierung von Affenkopf und Zylinder) und dann an verschiedenen Positionen mittels Transformationen mehrfach darzustellen, ohne die Geometriedaten zu duplizieren.