

# Szenengraphen für interaktive 3D-Anwendungen am Beispiel X3DOM



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Visual Computing

Winter Semester 2023-2024

**Prof. Dr. A. Kuijper**

Mathematical and Applied Visual Computing (MAVC)

Graphisch-Interaktive Systeme (GRIS)

Fraunhofer IGD

Fraunhoferstrasse 5

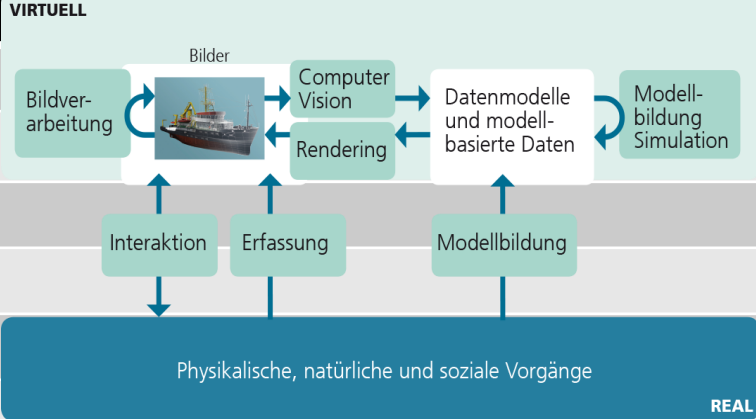
D - 64283 Darmstadt

E-Mail: [office@gris.tu-darmstadt.de](mailto:office@gris.tu-darmstadt.de)

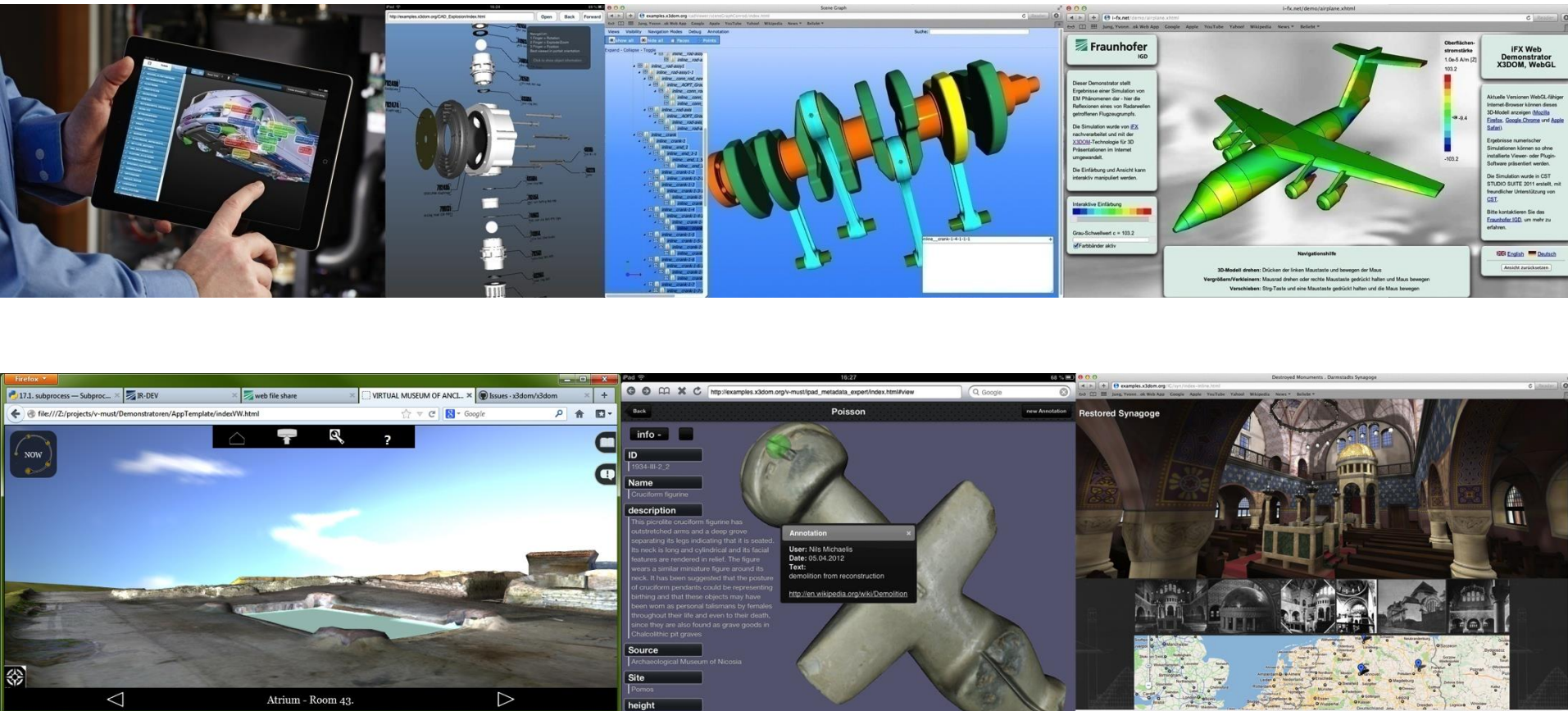
<http://www.gris.tu-darmstadt.de>

---

# Semesterplan

Datum		
20. Okt	Einführung + <a href="#">Visual Computing</a>	
27. Okt	<a href="#">Wahrnehmung</a>	
03. Nov	<a href="#">Objekterkennung</a> und <a href="#">Bayes</a>	
10. Nov	<a href="#">Fourier Theorie</a>	
17. Nov	<a href="#">Bilder</a>	
24. Nov	<a href="#">Bildverarbeitung</a>	
01. Dez	<a href="#">Grafikpipeline</a> & <a href="#">Eingabemodalitäten</a> & <a href="#">VR+AR</a>	
08. Dez	<a href="#">Transformationen</a> & <a href="#">2D/3D Ausgabe</a>	
15. Dez	<a href="#">3D-Visualisierung</a>	
<b>12. Jan</b>	<a href="#">X3D – 3D in HTML</a>	
19. Jan	Informationsvisualisierung	
26. Jan	User Interfaces + Multimedia Retrieval	
02. Feb	Farbe	
09. Feb	Biometrie (?)	

# Motivation: Anwendungen der 3D-Computergrafik



# Motivation: Anwendungen der 3D-Computergrafik

- 3D-Computergrafik ist bereits integraler Bestandteil vieler Anwendungen
- Technische Herausforderungen:
  - **Effizienz** der 3D-Anwendungen (Große Datenvolumen, Spiele, ...)
  - **Portabilität** (z.B. für Desktop- und Mobilgeräte)
- Weitere Herausforderungen:
  - Nahtlose **Integration** von 3D-Grafik in bestehende Anwendungen
  - **Vereinfachung und Vereinheitlichung** der Entwicklungsarbeit

→ Wir beschäftigen uns mit den letzten Punkten!



# Strukturierung von 3D-Szenendaten: Szenengraphen

- Die Bildgenerierung (Rendering) zur Darstellung einer 3D-Szene benötigt vielfältige **Informationen über die Szene** – z.B.?



# Strukturierung von 3D-Szenendaten: Szenengraphen

- Benötigte Informationen für das Rendering einer 3D-Szene:
  - Objekt-**Geometrie** (*z.B. für eine Säule*)
  - **Transformationen** (*z.B. für die Positionierung einzelner Säulen*)
  - **Materialien** (*Welche Farbe hat ein Objekt? Textur-Bilder?*)
  - **Kameras** (*Vordefinierte Ansichten, Kontrolle der Kamera, ...*)
  - **Lichter** (*Versch. Arten von Lichtquellen, Farben, ...*)
  - **Spezial-Effekte** (*Nebel, Schatten, Skyboxes, ...*)
  - ...

# Strukturierung von 3D-Szenendaten: Szenengraphen

- **Komplexe Beziehungen** zwischen Daten einer Szenen – z.B.:
  - Versch. Objekte verwenden das gleiche Material (Farbe, Textur)
  - Das gleiche Objekt wird an mehreren Orten instanziiert
  - Transformationshierarchie, Gruppieren von Objekten
  - ...

→ **Szenengraphen** strukturieren diese Information!

# Weiteres Beispiel / Demonstration: Auto-Konfigurator



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

## Car configuration prototype

This is a simple application prototype, which shows how you can change elements of the scene to configure a product. The original content and application idea of this example is taken from [Bitmanagement](#).

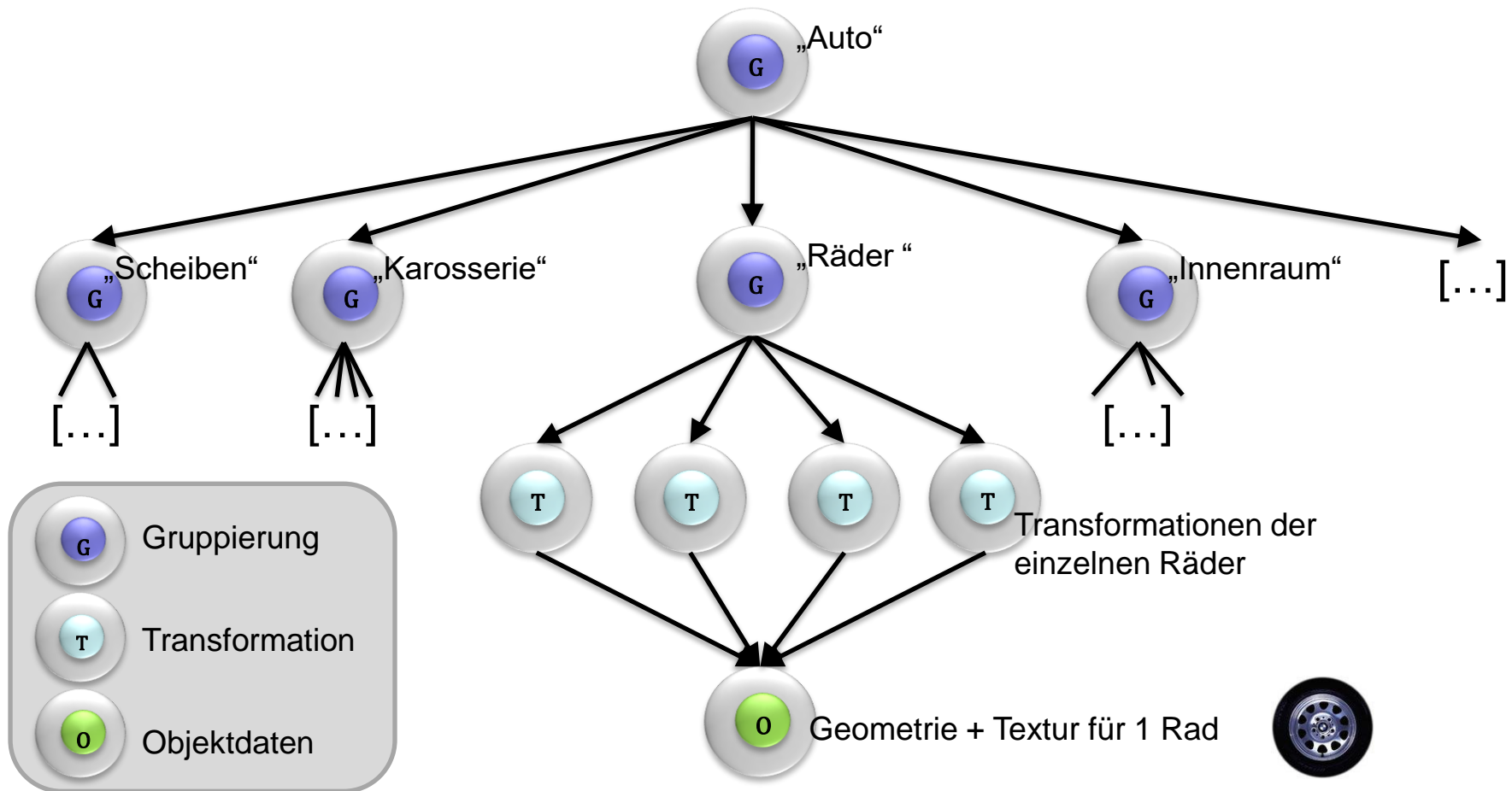




# Strukturierung von 3D-Szenendaten: Szenengraphen

- Wie kann diese Beispielszene **strukturiert** werden?
  - Das Auto ist **ein Objekt**, welches aus **vielen Einzelteilen** besteht
  - Verschiedene Gruppen von Teilen (z.B. Karosserie-Teile, Scheiben, Räder) verwenden das **gleiche Material / die gleiche Textur**
  - Alle Räder verwenden die **identische Geometrie**

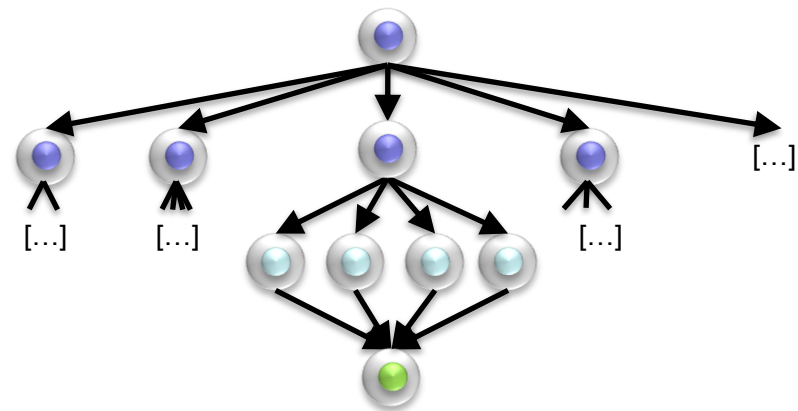
# Strukturierung von 3D-Szenendaten: Szenengraphen



Ein (vereinfachter) Beispiel-Szenengraph für unser Auto

# Strukturierung von 3D-Szenendaten: Szenengraphen

- Szenengraph ist ein **gerichteter, azyklischer Graph**
  - Engl.: **Directed Acyclic Graph (DAG)**
  - **Gerichtet:** Jede Kante hat eine Richtung
  - **Azyklisch:** Es gibt keine Zyklen („Rundwege“) im Graph
  - **Zusätzlich:** Szenengraph hat einen **Wurzelknoten**
  - Unterschied zu Bäumen: **Mehrere Elternknoten** möglich (Ausnahme: Wurzel)



# Strukturierung von 3D-Szenendaten: Szenengraphen

- Durchlaufen („Traversierung“) des Szenengraphs zum Rendering
  - Anwendung startet an der Wurzel
    - Jeder Kindknoten wird rekursiv abgearbeitet („traversiert“)
  - Wenn der gesamte Graph traversiert wurde, ist das Bild fertig!
    - (Daher: keine Zyklen im Szenengraph erlaubt)
- Konkrete Operationen während der Traversierung hängen vom Typ des jeweiligen Knotens ab (Beispiel: Nächste Folie)
- Während der Traversierung: Update verschiedener Zustände
  - Z.B. Transformationsmatrix („Current Transformation Matrix“, **CTM**)

# Strukturierung von 3D-Szenendaten: Szenengraphen

- Beispiel-Operationen für unsere Beispiel-Knotentypen:



- Gruppierung: Prüfe, ob aktuelle Gruppe eingeschaltet ist
  - Falls Ja: Traversiere Kindknoten, Sonst: Tue nichts



- Transformation: Anwendung einer Transformationsmatrix  $M$ 
  - CTM für die Kindknoten ist Produkt der CTM mit  $M$

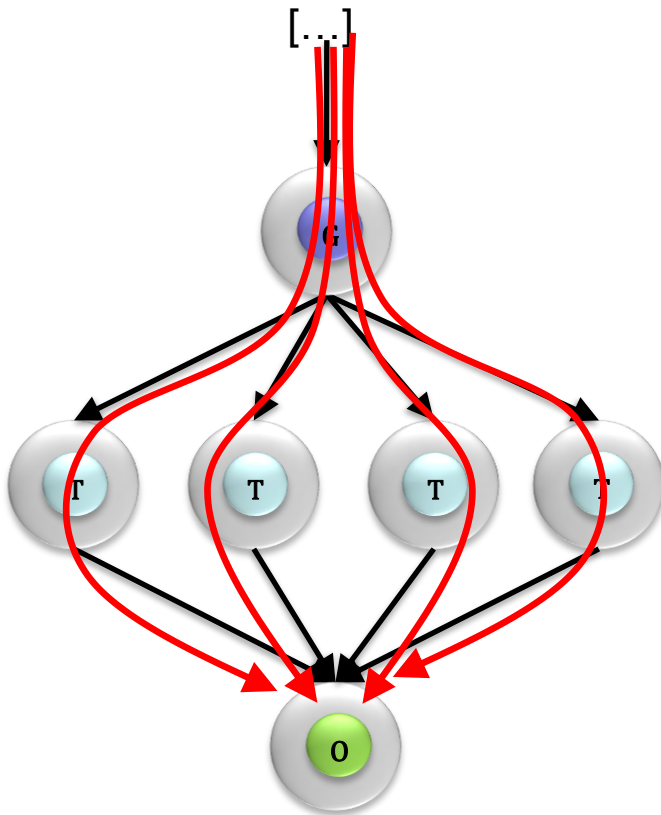


- Objektdaten: Zeichne Objekt
  - Verwende dabei die CTM

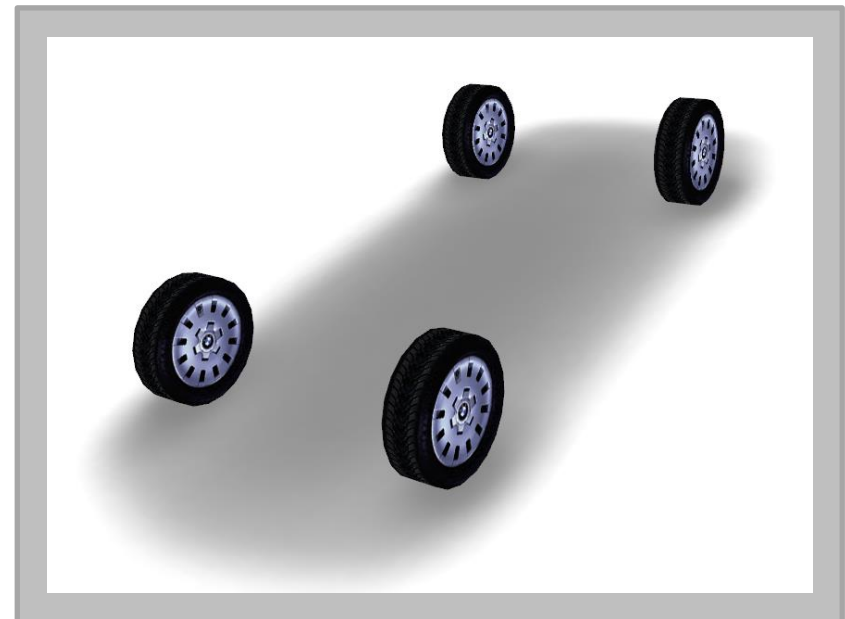


# Strukturierung von 3D-Szenendaten: Szenengraphen

- Beispiel: Zeichnen der Räder durch Traversierung des Graphen
  - Anpassung der CTM durch die Transformations-Knoten

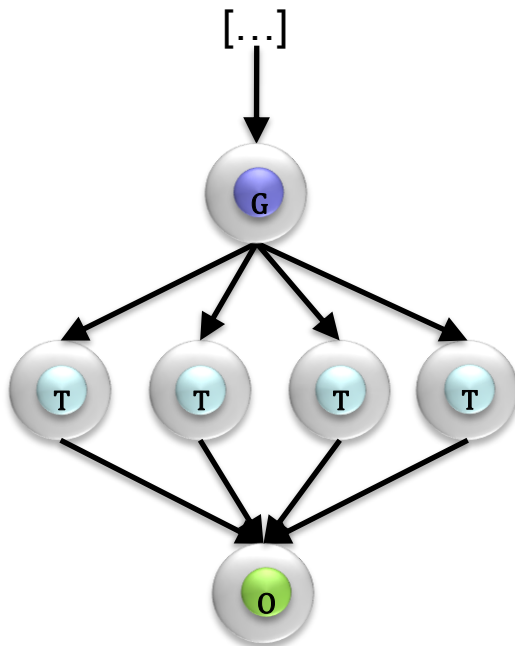


Ergebnisbild:



# Strukturierung von 3D-Szenendaten: Szenengraphen

- Vorteile des Szenengraph-Konzeptes:



- **Wiederverwendbarkeit** der Objektdaten  
(z.B. leichtes Austauschen der Felgen)
- **Semantische Gruppierung** der Objektdaten  
(z.B. einfaches Ein- / Ausblenden aller Räder)
- **Transformationshierarchie** ermöglicht Transformation von kompletten Gruppen, ohne diese explizit ändern zu müssen  
(z.B.: Transformation des gesamten Autos)

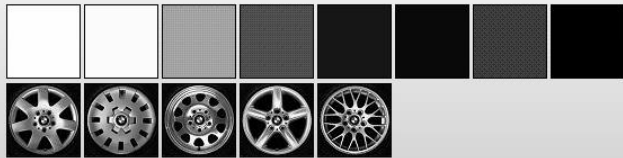
# Szenegraphen-API am Beispiel X3DOM



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

## Car configuration prototype

This is a simple application prototype, which shows how you can change elements of the scene to configure a product. The original content and application idea of this example is taken from [Biltmanagement](#).



Wie funktioniert das konkret?



# Szenegraphen-API am Beispiel X3DOM

- X3DOM = **Deklarative** Szenengraph-API auf Basis von **X3D** im **DOM**
  - **Deklarativ:** Szenegraph wird durch **strukturiertes Textformat** (z.B. XML) beschrieben (vergleichbar z.B. zu HTML)
  - **X3D:** Szenengraphen-Standard
    - XML-basiert, Nachfolger von VRML (nächste Folie)
    - Benötigt traditionell sog. X3D-Player (spezielle Anwendung)
  - **DOM:** HTML Document Object Model
    - Dokumenten-Baumstruktur und API in HTML / Javascript

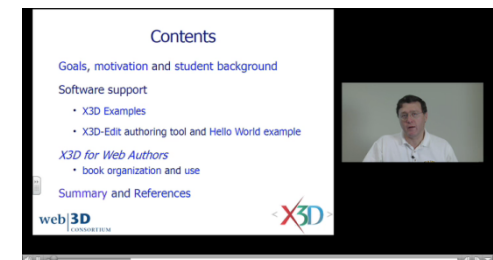
# Szenegraphen-API am Beispiel X3DOM

- Hintergrund: Geschichte der Standards VRML und X3D
  - **VRML (Virtual Reality Modeling Language)**
    - 1994 entwickeltes Konzept für Virtual Reality (VR) im **WWW**
    - Konzept stark angelehnt an **OpenInventor**, aber **deklarativ**
    - 1997 Veröffentlichung von **VRML 2.0** (auch: VRML97)
  - **X3D (Extensible 3D)**
    - Erlaubt **XML**, **Binary XML** oder **VRML** Encoding
    - Verschiedene neue **Nodes**
    - **Profile** (Gruppen von Nodes) für diverse Anwendungsfelder (z.B.: CAD, Geospatial, Humanoid Animation, ...)



# Szenegraphen-API am Beispiel X3DOM

- Einige Dokumentations-Ressourcen zu **X3D**
  - **Buch:** *X3D: Extensible 3D Graphics for Web Authors*
    - Gesamtüberblick, von Autoren des Standards
- **Online-Kurs** von Don Brutzman (NPS)
  - <https://www.movesinstitute.org/Video/Courses/X3dForWebAuthors/X3dForWebAuthorsVideo.html>
  - Folien ebenfalls online
- <http://www.web3d.org/realtime-3d/>
  - Zahlreiche **weitere Links** zu **Dokumentation** und **Beispielen**

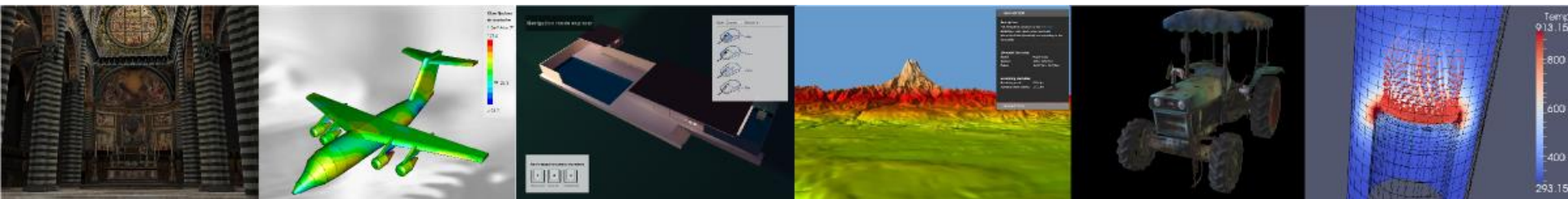


# Szenegraphen-API am Beispiel X3DOM

- 2009: Vorstellung von **X3DOM**
  - Integration von **X3D** in **(X)HTML**
    - verwendet X3D in **XML Encoding**
    - Anpassung einzelner X3D-Konzepte für die Browser-Umgebung (z.B. **HTML Mouse Events** auf 3D-Objekten)
    - Definiert **HTML-Profil** (Untermenge von Nodes) für X3D
- X3D / X3DOM bislang von keinem Browser nativ implementiert
  - Daher: „Polyfill“-Layer, Anwendungslogik in **JavaScript**
  - Verschiedene Rendering-Backends:  
X3D Browser-Plugin, Flash/Stage3D oder **WebGL**

# Szenegraphen-API am Beispiel X3DOM

- Wichtige Links zu **X3DOM**
  - **Dokumentations-Portal:** <http://doc.x3dom.org/>
    - Getting Started Guide
    - Umfassende Tutorials
    - Dokumentation sämtlicher verfügbarer Nodes
  - **Beispiel-Portal:** <http://examples.x3dom.org/>
    - Zahlreiche Anwendungsbeispiele



# Szenegraphen-API am Beispiel X3DOM



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
<!DOCTYPE html >
<html >
  <head>
    <link rel='stylesheet' type='text/css'
          href='http://www.x3dom.org/x3dom/release/x3dom.css' />
    <script type='text/javascript'
          src='http://www.x3dom.org/x3dom/release/x3dom.js'></script>
  </head>
  <body>
    <h1>Hello X3DOM World</h1>
    <x3d width='400px' height='400px'>
      <scene>
        <shape>
          <appearance>
            <material></material>
          </appearance>
          <box></box>
        </shape>
      </scene>
    </x3d>
  </body>
</html>
```

**Einbinden der Javascript-  
Library („Polyfill“)**

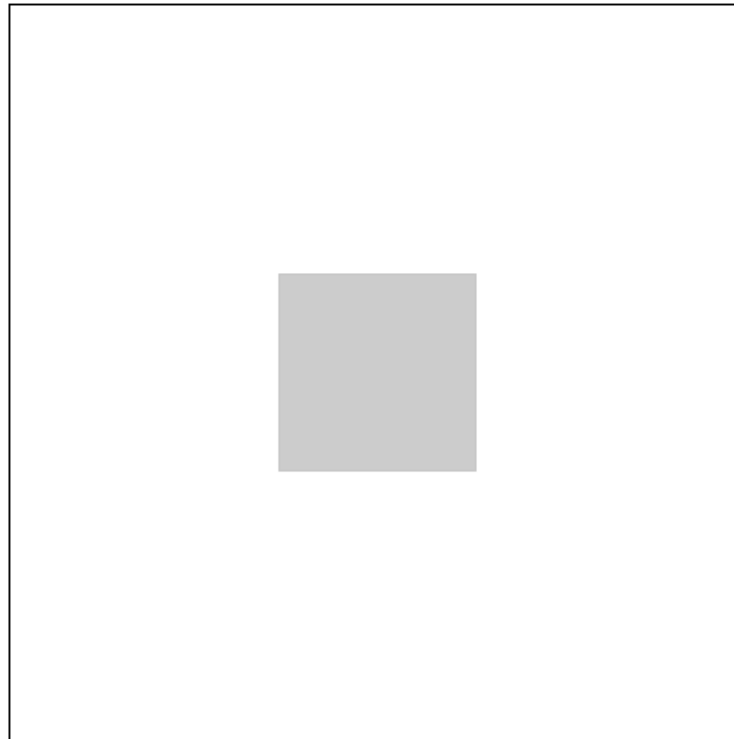
## Definition der X3D-Szene in HTML

(Achtung: z.Zt. Keine self-closing tags möglich,  
daher nicht 100% identisch mit X3D in XML)

# Szenegraphen-API am Beispiel X3DOM

- Hello, X3DOM

Hello X3DOM World





# Szenegraphen-API am Beispiel X3DOM

- Hello, X3DOM – Wie ist die Szene aufgebaut?

```
<x3d width='400px' height='400px'>
```

```
<scene>
```

**Deklaration des X3D-Kontext**  
(Elemente „X3D“ und „Scene“)

```
<shape>
```

```
<appearance>
```

```
<material></material>
```

```
</appearance>
```

**Definition der eigentlichen  
Szeneninhalte**  
(Elemente = „Nodes“)

```
<box></box>
```

```
</shape>
```

```
</scene>
```

**Deklaration des X3D-Kontext**  
(Elemente „X3D“ und „Scene“)

```
</x3d>
```

# Szenegraphen-API am Beispiel X3DOM

- Hello, X3DOM – Wie ist die Szene aufgebaut?

```
<shape>  
  <appearance>  
    <material></material>  
  </appearance>  
  <box></box>  
</shape>
```

# Szenegraphen-API am Beispiel X3DOM

- Einzelne Knoten der Beispielszene
  - **Shape:** Definiert ein **zeichenbares Objekt**
    - Shape = **Appearance** plus **Geometry**
  - **Appearance:** „**Erscheinungsbild**“ eines Objektes
    - Enthält Informationen über **Materialfarbe, Texturen, ...**
  - **Material:** Definiert Materialeigenschaften gem. **Phong-Modell**
  - **Box:** Spezieller **Geometry**-Knoten
    - Definiert Geometrie einer einfachen Box

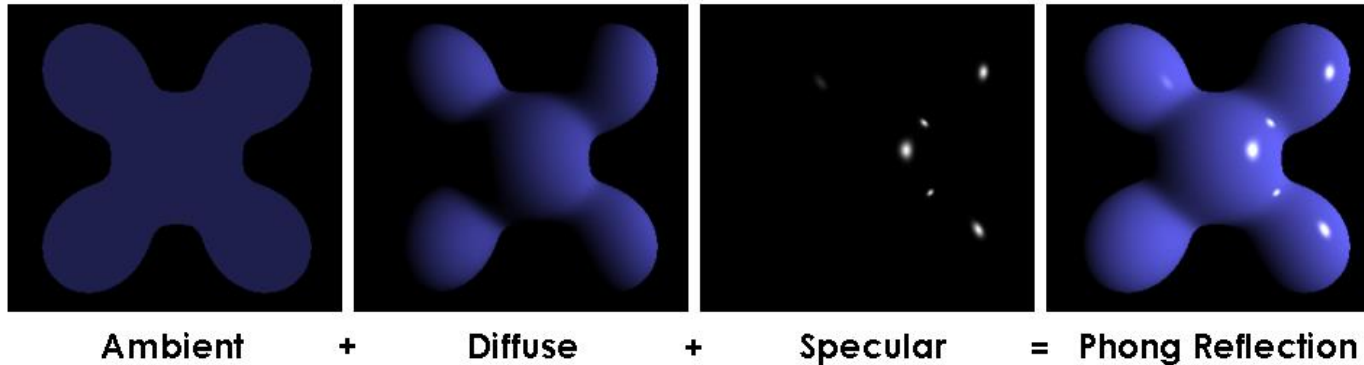
# Szenegraphen-API am Beispiel X3DOM

- Beispiel: Definition eines Materials mit den Default-Werten
  - Äquivalent zu `<material></material>`

```
<material  
  ambientIntensity='0.2' diffuseColor='0.8  
0.8 0.8' emissiveColor='0 0 0'  
  shininess='0.2' specularColor='0 0 0'  
  transparency='0'>  
</material>
```

# Szenegraphen-API am Beispiel X3DOM

- Materialeigenschaften modelliert gemäß **Phong-Modell**  
(→ Vorlesung Grafikpipeline)



- Beispiel:
  - Veränderung der diffusen Farbe unserer Box auf **rot**
  - Hinzufügen eines weißen Specular-Anteils



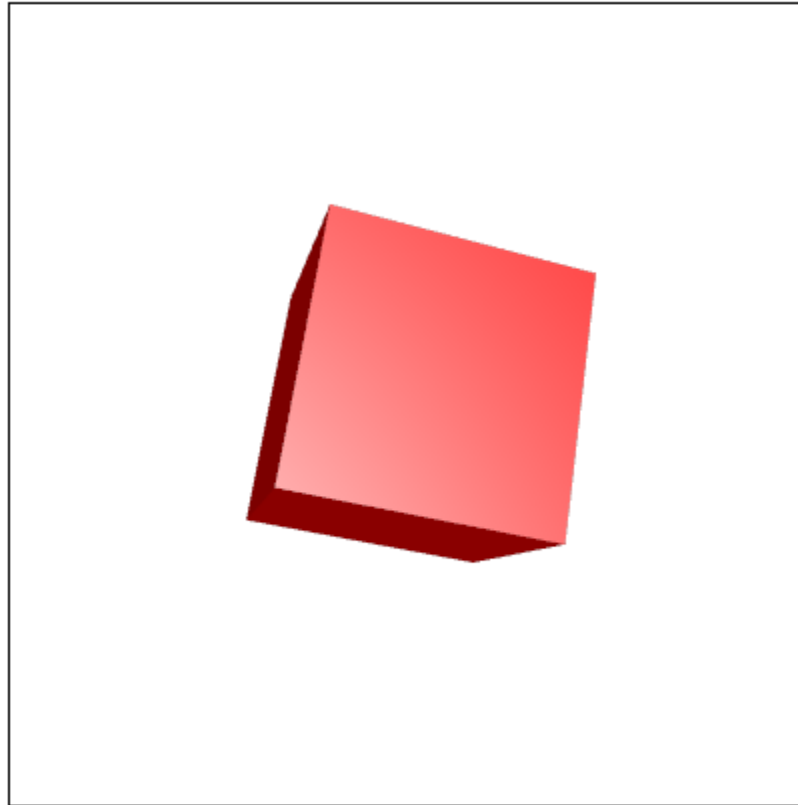
# Szenegraphen-API am Beispiel X3DOM

- Beispiel:
  - Veränderung der diffusen Farbe unserer Box auf **rot**
  - Hinzufügen eines weißen Specular-Anteils

```
<material diffuseColor ='1.0 0.0 0.0'  
           specularColor='1.0 1.0 1.0'>  
</material>
```

# Szenegraphen-API am Beispiel X3DOM: Demonstration

Hello X3DOM World



# Szenegraphen-API am Beispiel X3DOM



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- Kann man auch komplexere Objekte als eine Box zeichnen?
  - Spezielle Geometrie-Knoten wie Box, Sphere, Cone
  - Allgemeine Definition von 3D-Objekten über Polygone
  - Z.B.: **IndexedTriangleSet**-Knoten

```
<indexedtriangleset index='0 1 2 0 2 3 0 3 ...'>  
  <coordinate point='0.49 1.31 0.58 ...'></coordinate>  
  <normal vector='0.70 -0.22 0.67 ...'></normal>  
</indexedtriangleset>
```

# Szenegraphen-API am Beispiel X3DOM

```
<indexedtriangleset index='0 1 2 0 2 3 0 3 ...'>  
  <coordinate point='0.49 1.31 0.58 ...'></coordinate>  
  <normal vector='0.70 -0.22 0.67 ...'></normal>  
</indexedtriangleset>
```

Hello X3DOM World



# Szenegraphen-API am Beispiel X3DOM




TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
<indexedtriangleset index='0 1 2 0 2 3 0 3 ...'>  
  <coordinate point='0.49 1.31 0.58 ...'></coordinate>  
  <normal vector='0.70 -0.22 0.67 ...'></normal>  
</indexedtriangleset >
```

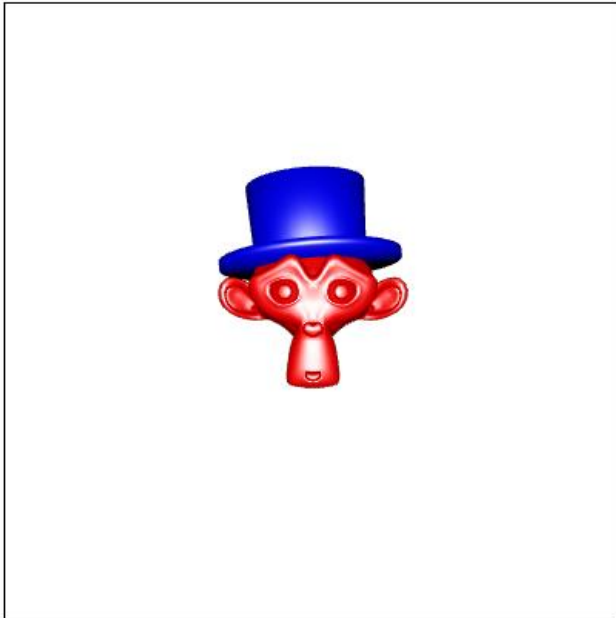
- Definition der 3D-Daten im DOM problematisch für große Modelle
  - Lange Ladezeiten, unübersichtlich, schwer zu editieren
- Lösung: Auslagern der 3D-Daten in externe Binärdateien
  - Prinzip analog zu Bildern in HTML (**<img src='foo.png'>**)
  - Mehr dazu z.B. unter <http://www.x3dom.org/pop/> und <http://www.x3dom.org/src/>, sowie in diversen Web3D-Papers
- Hier: der Einfachheit halber Verwendung von IndexedTriangleSet

# Szenegraphen-API am Beispiel X3DOM

- Transformation von Objekten?
  - **Transform-Knoten**
    - erlaubt Translation, Rotation, Skalierung und Scherung (s. Vorlesung zu Transformationen)
  - Schachteln von Transform-Knoten erlaubt Aufbau einer **Transformationshierarchie** 
  - **Beispiel:** Suzanne mit Hut (nächste Folie)


# Szenegraphen-API am Beispiel X3DOM

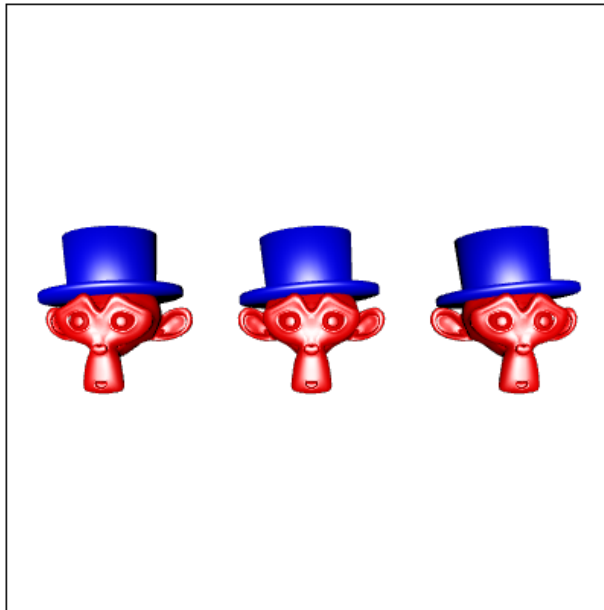
- 2 **Shape**-Knoten (Suzanne + rotes Material, Hut + blaues Material)
  - **Group**-Knoten fasst alles zu einem neuem Objekt zusammen
  - **Transform**-Knoten verschiebt Hut



```
<group>  
  <shape>...</shape>  
  <transform  
    translation='0 1.5 0'>  
    <shape>...</shape>  
  </ transform >  
</group>
```

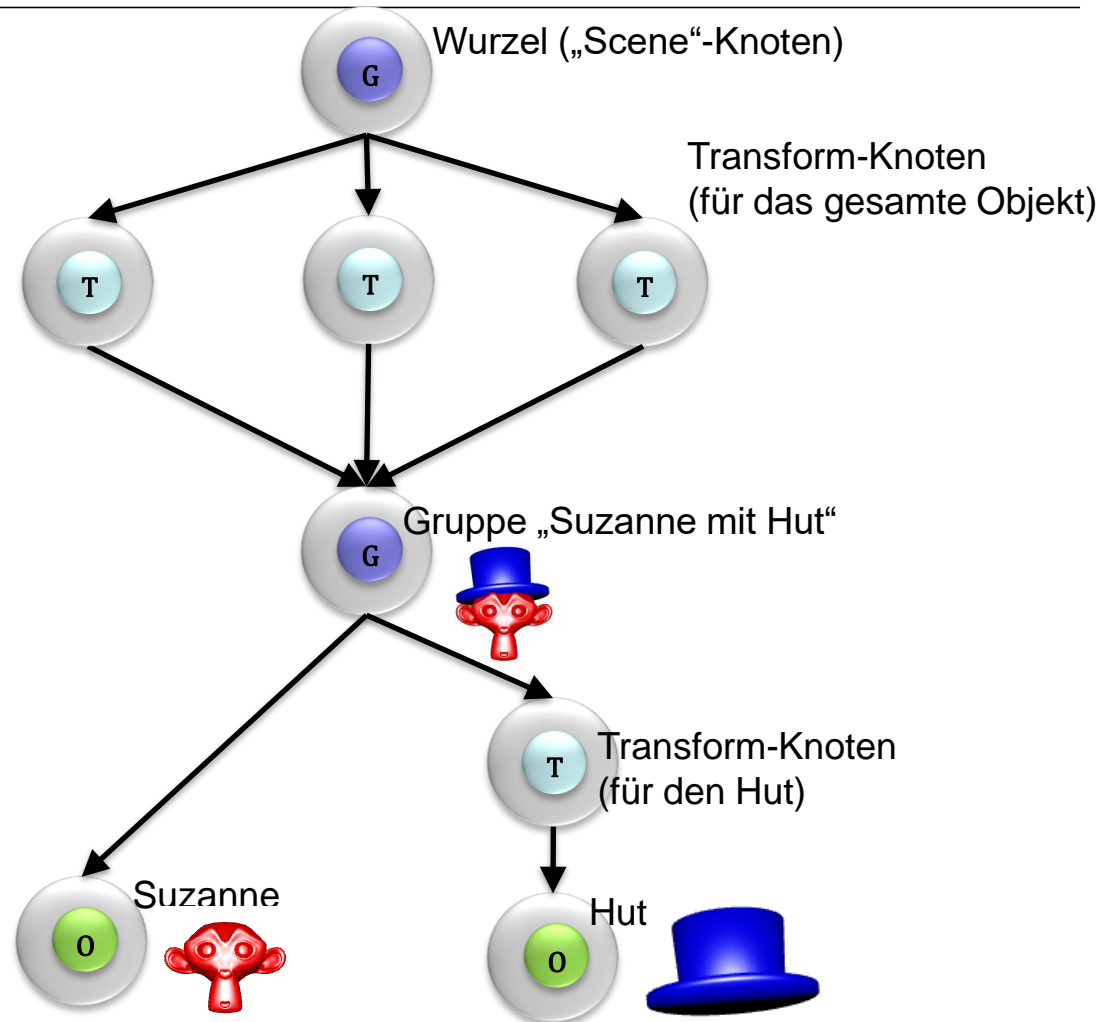
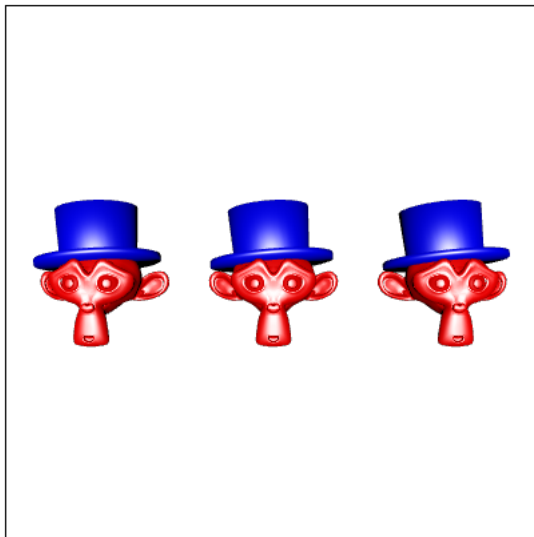
# Szenegraphen-API am Beispiel X3DOM

- Frage: Wie kann ich **mehrere Instanzen** zeichnen?
  - So wie in unserem Beispiel mit den Rädern des Autos 
  - Ich möchte das Objekt nur **einmal** definieren!
  - Anwendung der anfangs diskutierten Szenengraph-Konzepte



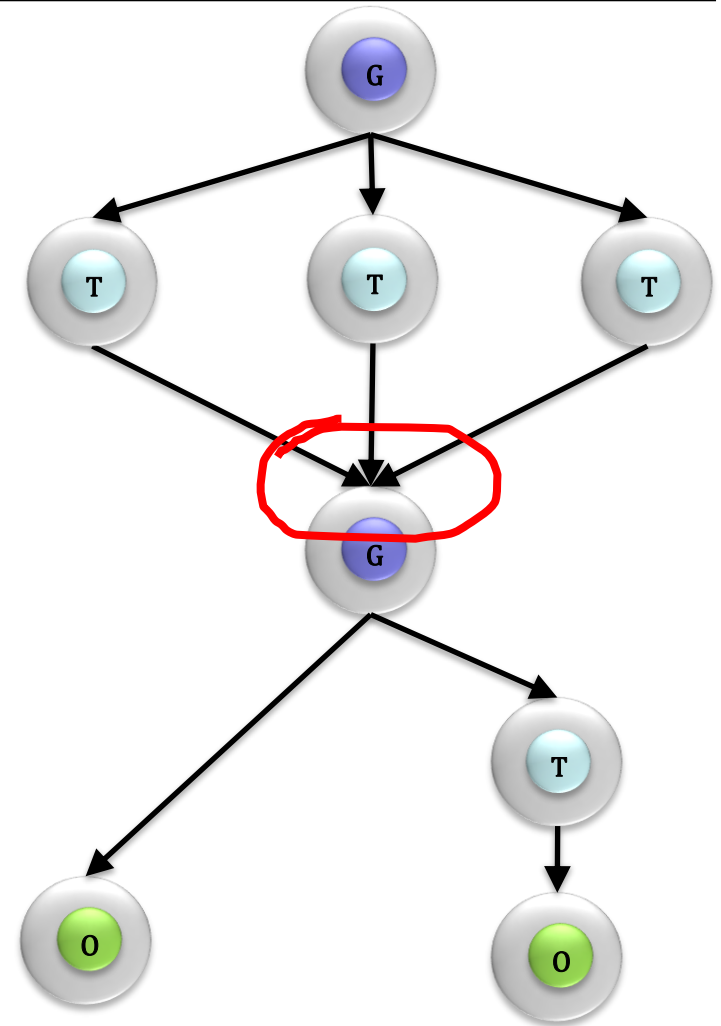


# Szenegraphen-API am Beispiel X3DOM



# Szenegraphen-API am Beispiel X3DOM

- **Problem** bei der Realisierung X3DOM (d.h. mittels HTML):
  - HTML-Elemente haben immer nur **ein einziges Elternelement**
  - HTML DOM ist ein **Baum**, kein Graph!
  - Wie können wir damit trotzdem unseren Szenengraphen umsetzen?
- **Lösung:** X3D definiert **DEF/USE-Mechanismus**
  - Definition eines Knotens mittels **DEF='...'**
  - Wiederverwendung mit anderem Elternknoten durch Platzhalter-Kindknoten mit Verweis **USE='...'**

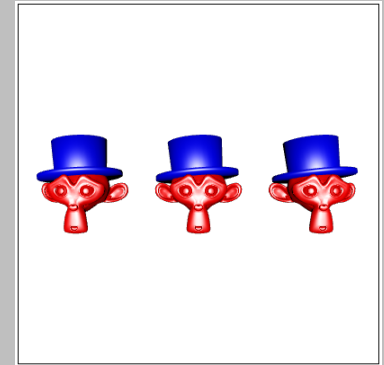


# Szenegraphen-API am Beispiel X3DOM




TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
<group DEF='SUZANNE_WITH_HAT'>  
  ...  
</group>  
<transform translation='-3.5 0 0'>  
  <group USE='SUZANNE_WITH_HAT'></group>  
</transform>  
<transform translation='3.5 0 0'>  
  <group USE='SUZANNE_WITH_HAT'></group>  
</transform>
```



# Szenegraphen-API am Beispiel X3DOM

- X3DOM – nächste Schritte 
  - **Viewpoint**-Knoten zur Definition einer Kameraposition
  - **DirectionalLight**-Knoten für einfache, direktionale Beleuchtung
  - **ImageTexture**-Knoten zur Einbindung von Texturbildern
  - ...
- Tipps
  - X3D-Tutorials anschauen / X3D-Buch lesen
  - Tutorials auf <http://doc.x3dom.org/>
  - Online-Beispielanwendungen anschauen (HTML-Quellcode)
  - Übung zur Vorlesung 😊

- Szenengraphen helfen uns, die 3D-Szenendaten zu **strukturieren**
- Ein Szenengraph ist ein gerichteter, azyklischer Graph (**DAG**)
  - **Wiederverwendbarkeit** von Knoten
  - Semantische **Gruppierung**
  - **Transformationshierarchien**
- **X3DOM** ist ein deklarativer 3D-Szenengraph in HTML
  - Basiert auf **X3D**, einem Szenengraphen-Standard
  - Knoten als HTML DOM-Elemente
  - DEF/USE zur Realisierung von Graphenstrukturen

# Vielen Dank für die Aufmerksamkeit