X3D-Format

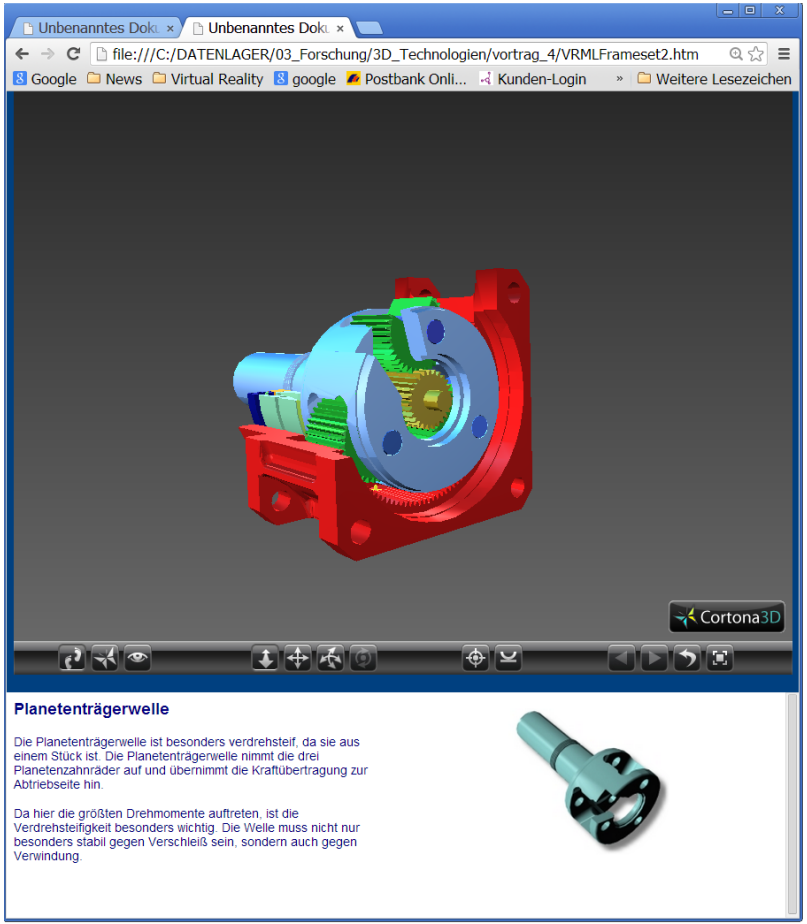
X3D: 3D-Modelle im Browser nativ visualisieren

Prof. Dipl.-Ing. Martin Schober, Hochschule Karlsruhe – Technik und Wirtschaft

1996: VRML-Visualierung durch Browser + Plugin

Das VRML-Format konnte schon vor 1996 als 3D-Modell mit Browser visualisiert und mittels JavaScript mit Interaktivität ausgestattet werden.

VRML = Virtual Reality Modeling Language



VRML: Link mit URL und Beschreibung

```
children [
      DEF __38C808E_8 Transform {
      scaleOrientation -0.369 -0.137 0.147 0.868
       children [
              Anchor {
                url "../ptw.html"
                parameter "target=showframe"
                description "Planetentraegerwelle"
```

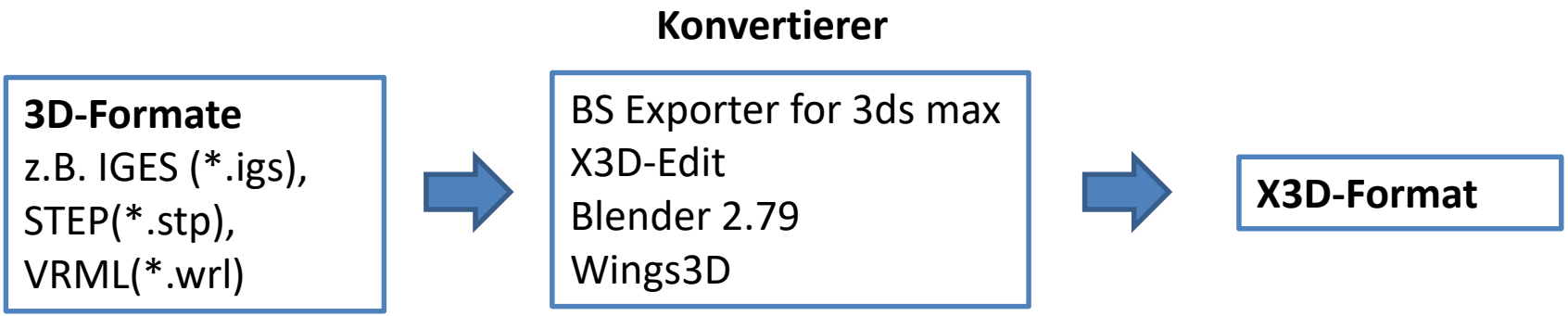
2001: X3D

- Extensible 3D oder kurz X3D, ist eine auf XML basierende Beschreibungssprache für 3D-Modelle.
- X3D wird betreut durch das Web3D-Konsortium und wurde 2001 vom W3C-Konsortium als offizieller Standard für 3D-Inhalte im Internet verabschiedet.
- X3D ist seit Dezember 2004 als ein offener ISO-Standard spezifiziert.
- Genau wie in VRML lassen sich auch in X3D dreidimensionale virtuelle Welten, Spiele, wissenschaftliche Visualisierungen und interaktive Lernanwendungen in Echtzeit realisieren.

  Gegenüber VRML stehen jedoch bei X3D wesentlich mehr standardisierte Möglichkeiten und Schnittstellen bereit.

Quelle: Wikipedia

Werkzeuge um das X3D-Format zu erzeugen



Visualisierung von X3D-Daten im Browser

X3DOM-Framework des Frauenhofer Instituts

Bei dem X3DOM-Framework des Frauenhofer Instituts handelt es sich um eine Technologie, die es ermöglicht X3D-Inhalte ohne zusätzliche Plug-Ins innerhalb eines WebGL fähigen Webbrowsers lauffähig darzustellen.

X3DOM basiert auf JavaScript und wird in der HTML-Instanz, in der das X3D-Element angezeigt werden soll, eingebunden.

```
<script type="text/javascript" src="js/x3dom.js" />
```

JavaScript-Link für das X3DOM-Framework im Head-Bereich der HTML-Instanz

Visualisierung von X3D-Daten im Browser

Außerdem wird noch ein CSS-Stylesheet benötigt, welches ebenfalls vom Frauenhofer Institut kostenlos angeboten wird.

Dieses muss zusammen mit dem JavaScript in die jeweilige HTML-Instanz eingebunden werden.

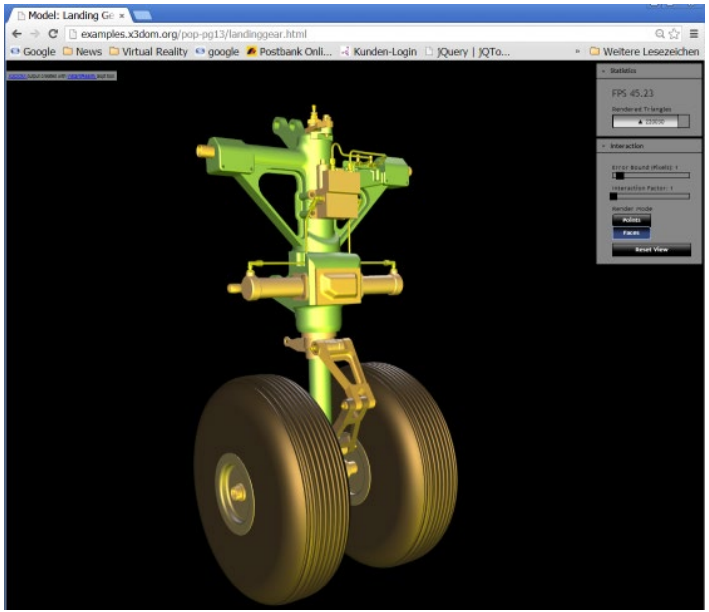
```
<link href="css/x3dom.css" rel="stylesheet" type="text/css" />
```

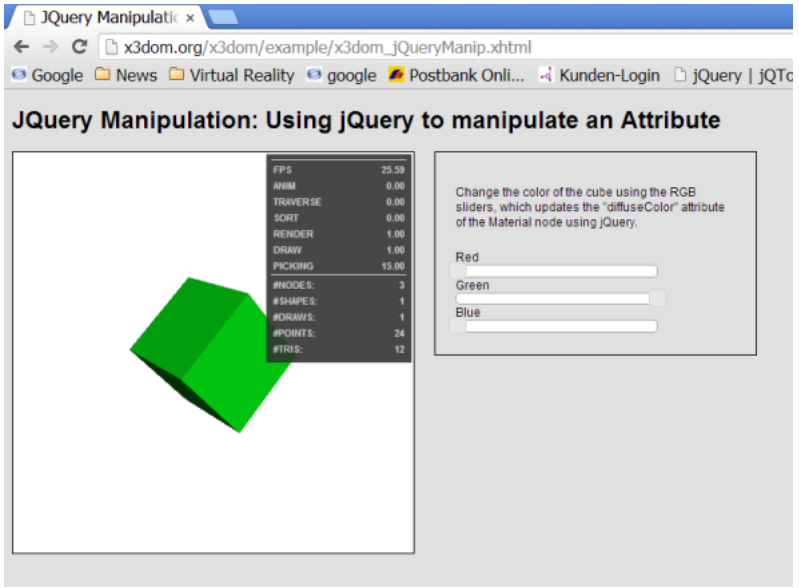
CSS-Stylesheet-Link für das X3DOM-Framework im Head-Bereich der HTML-Instanz

X3D-Beispiel

http://x3dom.org/x3dom/example/x3dom_objectAndText.xhtml

http://examples.x3dom.org/pop-pg13/landinggear.html





Blender Import- & Export-Formate

Außer dem Blender-Format .blend können folgende Dateiformate importiert und exportiert werden:

- Collada (Default) (.dae)
- Stanford (.ply)
- Stl (.stl)
- 3D Studio (.3ds)
- Autodesk FBX (.fbx)
- Wavefront (.obj)
- X3D Extensible 3D (.x3d)
- Motion Capture (.bvh)
- Scalable Vector Graphics (.svg)
- **X3D Extensible 3D (.wrl)**

Blender-Modelle für Browservisualisierung aufbereiten

Variante 1: X3D → Instantreality-Konverter → HTML5

Variante 2: X3D → als <inline> Element in eine HTML5-Datei einbinden.

Variante 3: X3D → in JavaScript/Three.js umwandeln.

Export von 3D-Modellen aus Blender, so dass sie in Websites eingebunden werden können
Ansprechen einzelner Elemente des 3D-Objekts.

Einbindung mit X3D (Variante 1)

Variante 1:

1. Export aus Blender in .x3d

2. Konvertieren der x3d-Daten mit dem Online-Konverter ***Instantreality*** vom Fraunhofer Institut

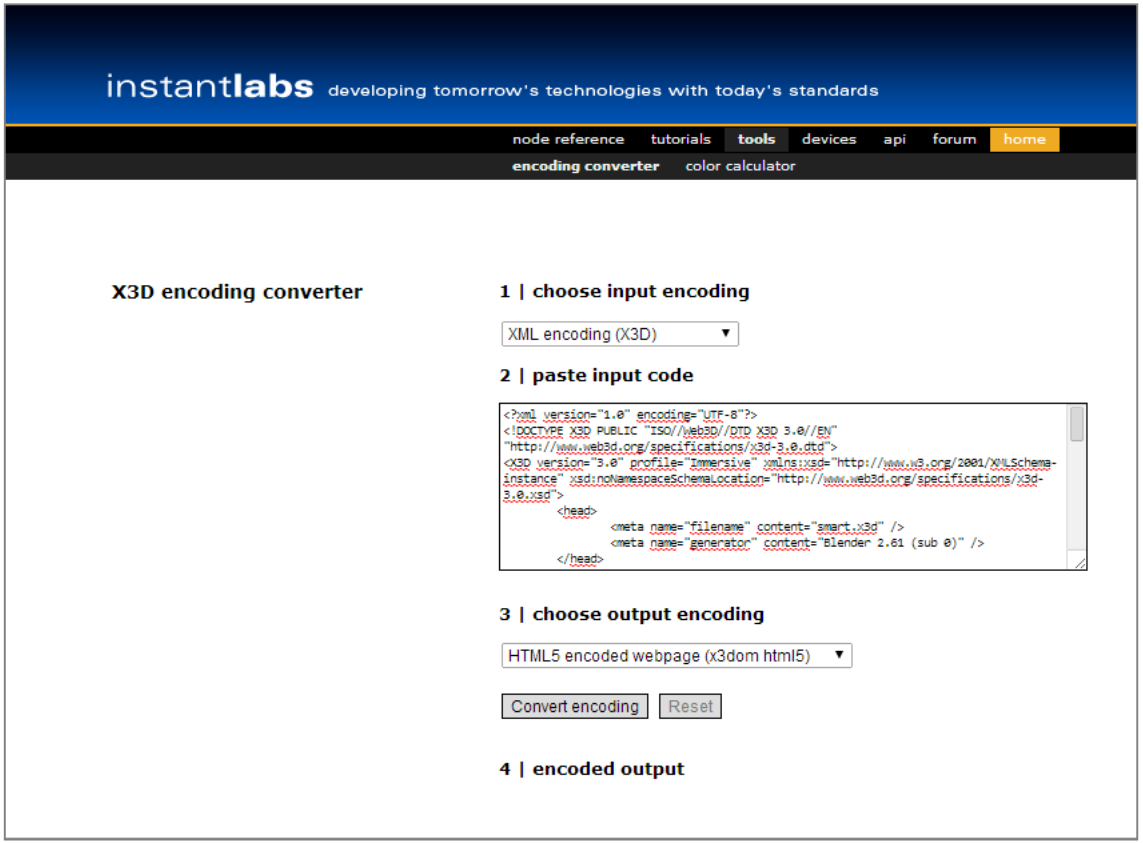
   http://doc.instantreality.org/tools/x3d_encoding_converter/

3. Speichern als HTML-Datei

Die erste Methode, 3D-Modelle mittels X3DOM im Browser darzustellen, ist sehr einfach und erfordert keine oder nur sehr geringe Erfahrung mit HTML, JavaScript oder sonstigen Skript- oder Programmiersprachen.

Dafür wird die Datei zunächst aus Blender in das X3D-Format exportiert.

Der Konvertierungsvorgang (Variante 1)



Sobald der Code konvertiert ist, kann er aus dem Encoded-output-Fenster kopiert, und mit der Endung html gespeichert werden.

Encoded output (Variante 1)

4 | encoded output

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv='Content-Type' content='text/html;charset=utf-8'></meta>
    <link rel='stylesheet' type='text/css' href='http://www.x3dom.org/x3dom/release/x3d
    <script type='text/javascript' src='http://www.x3dom.org/x3dom/release/x3dom.js'></
  </head>
  <body>
    <x3d id='someUniqueId' showStat='false' showLog='false' x='0px' y='0px' width='400p
      <scene DEF='scene'>
        <transform>
          <navigationInfo headlight='false'></navigationInfo>
          <pointLight DEF='LocLight' color='1 0.647 0.376' location='4.83 4.09 1.12'></
          <pointLight DEF='LocLight_1' color='0.00784 0.698 1' location='-3.63 3.05 5.1
          <pointLight DEF='LocLight_2' location='1.23 2.93 -5.39'></pointLight>
          <pointLight DEF='LocLight_3' intensity='0.5' location='0 2 -8.74e-08'></point
          <spotLight DEF='SpotLight' intensity='0.5' cutOffAngle='1.05' direction='0.498 -1.85 -0.581' location='-2.93 0.605 11.5'></spotLight>
          <pointLight DEF='LocLight_4' intensity='0.5' location='-2.31 0.407 10.5'></pointLight>
          <pointLight DEF='LocLight_5' intensity='0.5' location='0.923 0.332 9.07'></pointLight>
          <pointLight DEF='LocLight_6' intensity='0.41' location='0.351 -0.0432 9.86'></pointLight>
          <transform DEF='Amapi_scene' rotation='0.684 -0.646 -0.261 2.7' scaleOrientation='0.312 0.00789 0.109 0.673' translation='-0.596 1.73e-07 8.53'>
            <transform scaleOrientation='-0.369 -0.137 0.147 0.868'>
              <anchor description='Planetentraegerwelle' parameter='"target=showframe"' url='"../ptw.html"'>
                <transform DEF='__38C808E_8' scaleOrientation='-0.369 -0.137 0.147 0.868'>
                  <shape>
                    <appearance>
                      <material ambientIntensity='0.51' diffuseColor='0.145 0.757 0.212' shininess='0.48' specularColor='0.46 0.46 0.46'></material>
```

Lokale Konvertierung mit dem aopt-Konverter (Variante 1)

Die lokal Konvertierung geschieht mit dem Avalon-Optimizer der heruntergeladen und installiert werden muß.

http://www.instantreality.org/downloads/

Nun wird in der Windows-Eingabeaufforderung in den Ordner gewechselt, in dem sich die aopt.exe-Datei befindet (bei Windows meist C:\Program Files\Instant Reality\bin).

Anschließend wird der Befehl

```
aopt –i Dateiname.x3d –N Dateiname.html
```

eingegeben, um die Datei zu konvertieren.

Änderungen der 3D-Szene wie das Ändern der Hintergrundfarbe oder der Beleuchtung, können in der HTML-Datei vorgenommen werden.

X3D-Datei in HTML5-Datei einbinden (Variante 2)

Variante 2:

Erstellen einer HTML-Datei. Einbinden des 3D-Modells in einem <inline>-Element

Webserver nötig!!!

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8">
    <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>
<x3d id="3dInhalt" width="800px" height="600px">
   <scene>
     <inline url="smart.x3d" ></inline>
   </scene>
  </x3d>
</body>
</html>
```

Variante mit dem JS-Framework three.js (Variante 3)

Vorgehensweise

Um in Blender erstellte 3D-Modelle mittels three.js im Browser anzeigen zu können, muss die Datei in das Format Wavefront (.obj) exportiert werden.

1. Export aus Blender in .obj mit speziellen Exporteinstellungen!

2. Konvertieren in JavaScript mit dem Konvertierer AlteredQualia. Hierzu wird Python in einer Version 2.x benötigt. Konverter und die python.exe-Datei müssen im gleichen Verzeichnis abgespeichert sein.

```
python convert_obj_three.py -i Dateiname.obj -o Dateiname.js
```

Variante mit dem JS-Framework three.js (Variante 3)

3. Erstellen der 3D-Szene mit three.js

   Nun kann in einem Editor ein einfaches HTML-Grundgerüst erstellt werden.
   Im head wird die JavaScript-Datei three.js eingebunden.
   Im body wird innerhalb eines <script>-Tags die 3D-Szene erstellt
   Die .obj-Datei wird durch den JSONLoader eingebunden.

4. Einbindung des 3D-Modells

   Da hier JavaScript-Dateien in das HTML-Dokument eingebunden werden, muss die Datei auch hier über einen Webserver aufgerufen werden, um im Browserfenster korrekt angezeigt werden zu können.

→ Webserver nötig

Das X3D-Modell im Browser ausrichten über Viewpoint

```
<Viewpoint DEF="CA_Camera"
    centerOfRotation="0 0 -30"
    position="-0.00 -0.00 -0.00"
    orientation="-0.92 0.35 0.17 0.00"
    fieldOfView="0.858"
/>
```

Das Feld centerOfRotation gibt ein Zentrum an, um das das Objekt gedreht werden soll. Im vorliegenden Fall musste die Z-Koordinate um -30 verschoben werden, damit das Objekt auf der Stelle gedreht werden kann.

Die Positionsfelder des Knotens Ansichtspunkt geben eine relative Position im lokalen Koordinatensystem an. Die Position ist relativ zum Ursprung des Koordinatensystems (0,0,0). Damit kann das Objekt innerhalb des 3D-Fensters verschoben werden.

Die Orientierungsfelder des Knotens Ansichtspunkt legen die relative Orientierung zur Standardorientierung fest.

Bevorzugter minimaler Betrachtungswinkel von diesem Standpunkt aus in Bogenmaß. Kleines Sichtfeld entspricht etwa einem Teleobjektiv, großes Sichtfeld entspricht etwa einem Weitwinkelobjektiv. Tipp: Das Verändern des Blickpunktabstandes zum Objekt kann für das Zoomen besser geeignet sein.

Das Modell im Browser ausrichten über Viewpoint


```
<Viewpoint DEF="CA_CA_Camera"
                    centerOfRotation="0 0 -8.5"
                    position="-0.00 -0.00 0.00"
                    orientation="-0.87 -0.49 0.11 0.00"
                    fieldOfView="1.5"
                    />
```

Interaktivität integrieren

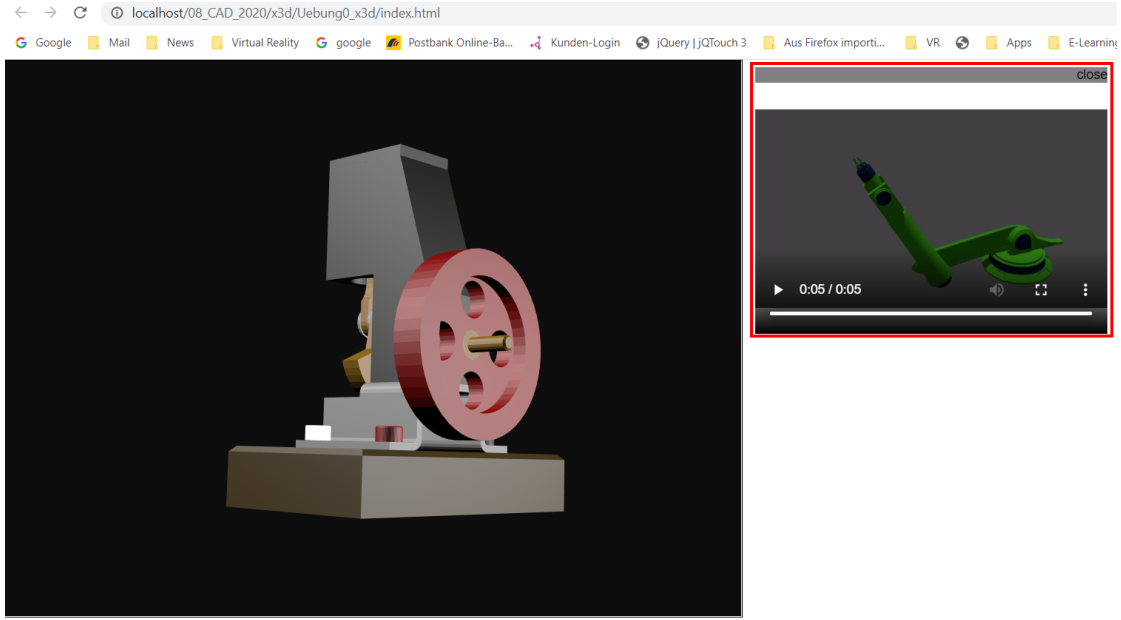
X3DOM-Knoten können ein onclick-Ereignis enthalten, das einen Funktionsaufruf ermöglicht. Jedes Mal, wenn Sie in die Szene klicken, werden Schnittstellentests mit Hilfe von Strahlen (Rays) und einem speziellen Puffer durchgeführt. Wenn ein Objekt erkannt wird, kann eine Funktion oder ein Befehl aufgerufen werden.

```
<Group DEF="group_ME_Volumenk16">
<Shape onclick="info('Volumenk16');">
    <Appearance>
        <Material USE="MA_mat_3" />
    </Appearance>
    <IndexedFaceSet solid="true"
```

Quelle: https://doc.x3dom.org/tutorials/animationInteraction/picking/index.html

Übung U5 zur Abgabe bis zum 3.5.2021 10:00 Uhr

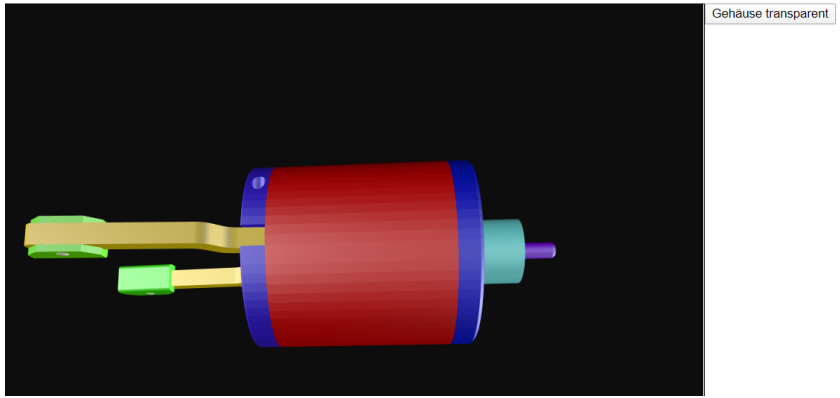
Bauen Sie mit dem x3d-Modell Druckluft.x3d eine HTML-Seite mit dem x3d-Modell, bei dem Bauteile geklickt werden können und in einem DIV-Container multimediale Informationen zu diesem Bauteil bereitgestellt werden. Die Daten zur Übung finden Sie in der zur Übung gehörenden ZIP-Datei.

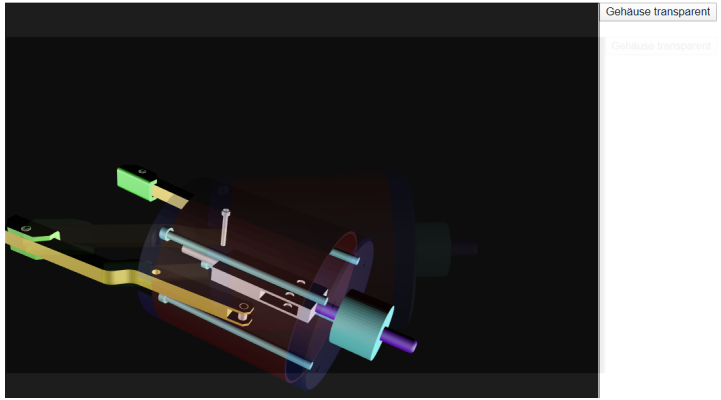


Das 3D-Modell interaktiv u. dynamisch verändern

Zum Einbinden von X3D-Dateien stellt X3DOM einen Inline-Knoten zur Verfügung. Durch die Verwendung von Inline im x3d-Kontext sind Sie in der Lage, externe x3d-Dateien zu laden.

Diese Dateien können einfache Objekte, komplexere Objekte oder sogar ganze Szenen enthalten. Die einzelnen Bauteile können über JS-Befehle dynamisch verändert werden.





Durch das Klicken auf eine Schaltfläche oder auf die Bauteile im Modell, können die Bauteile selbst verändert werden.

Das 3D-Modell interaktiv u. dynamisch verändern

nameSpaceName: Gibt den Namensraum des Inline-Knotens an.

mapDEFToID: Gibt an, ob der DEF-Wert als Id verwendet wird, wenn keine andere Id eingestellt ist.

getAttribute: Die Methode ermittelt den Wert eines bestimmten Attributs in einem Element.

setAttribute: Fügt dem angegebenen Element ein Attribut hinzu oder ändert den Wert eines vorhandenen Attributs.

DEF: DEF-Namen bieten ein Label für jeden Knoten in einem X3D-Modell ,einschließlich der untergeordneten Knoten, die diesen Untergraphen bilden. Er kann als Äquivalent zu ID-Attributen in XML oder HTML angesehen werden. Wie bei ID sollten DEF eindeutig sein, also nur einmal pro Modell verwendet werden.

Das 3D-Modell interaktiv u. dynamisch verändern

HTML

```
<x3d id="3dInhalt" width="800px" height="600px">
   <scene>
    <inline nameSpaceName="greifer" mapDEFToID="true" url="greifer.x3d" />
   </scene>
  </x3d>
```

JavaScript

```
function transp(part) {
  if(document.getElementById('greifer__' + part).getAttribute('transparency')!= '0.8')
     document.getElementById('greifer__' + part).setAttribute('transparency', '0.8');
  else
     document.getElementById('greifer__' + part).setAttribute('transparency', '0.0');
    }
```

X3D

```
<Group DEF="group_ME_DECKEL_1">
          <Shape onclick="transp('MA_Material13')">
             <Appearance>
                    <Material DEF="MA_Material13"
                     ... transparency="0.0"/>
```

Das 3D-Modell interaktiv u. dynamisch verändern

Um das Modell zu manipulieren, müssen wir sicherstellen, dass keine Namenskonflikte vorliegen. Deshalb wird ein nameSpace verwendet. Wie im vorherigen Beispiel X3DOM, HTML, CSS und JavaScript gezeigt, greift das Programm auf einen Knoten über seine ID zu. Um diese ID's zu erhalten, ist das Attribut mapDEFToID auf true zu setzen.

Jetzt kann auf jeden Knoten des Modells zugegriffen werden, indem das Namensraum-Präfix und ein doppelter Unterstrich ergänzt um den DEF des einzelnen Bauteils des x3d-Modells verwendet wird. Dies erlaubt es, die Transparenz oder andere Eigenschaften der Bauteile zu verändern. Hierzu kann eine JavaScript-Funktion, wie in der vorigen Folie gezeigt, verwendet werden.

Example

Bauteile dynamisch verändern (Farbe, Transparenz)

1. Einbinden der x3d-Datei in die HTML-Datei.

2. Im inline-Tag einen nameSpaceName vergeben und mapDEFToID auf true setzen!

```
<x3d id="3dInhalt">
 <scene>
  <inline nameSpaceName="Greifer" mapDEFToID="true" url="greifer.x3d"/>
 </scene>
</x3d>
```

Bauteile dynamisch verändern (**Farbe**, Transparenz)

3. Eine Funktion erstellen, die die **Farbe** des geklickten Bauteils auf die Farbe orange setzt.

```
function changecolor(Bauteil)
{
 if(document.getElementById('greifer__' + Bauteil).getAttribute('diffuseColor')!= '0.800
    0.400 0.110')
     document.getElementById('greifer__' + Bauteil).setAttribute('diffuseColor‘,
      '0.800 0.400 0.110');
 else
     document.getElementById('greifer__' + Bauteil).setAttribute('diffuseColor', '0 0
    0‘);
}
```

4. In der x3d-Datei das onclick-Ereignis und die Variablenübergabe einrichten.

```
<Shape onclick="changecolor('MA_Material12')">
```

Bauteile dynamisch verändern (Farbe, **Transparenz**)

5. Erstellen einer Funktion, um Bauteile transparent zu schalten.

```
function transp(part) {
   if(document.getElementById('greifer__' + part).getAttribute('transparency')!= '0.8')
       document.getElementById('greifer__' + part).setAttribute('transparency', '0.8');
   else
     document.getElementById('greifer__' + part).setAttribute('transparency', '0.0');
}
```

6. In der x3d-Datei das onclick-Ereignis und die Variablenübergabe einrichten.

```
<Shape onclick="transp('MA_Material10')">
```

Literatur und Links

- http://www.blender.org/about/, 12.03.2014
- http://t3n.de/news/3d-inhalt-html-syntax-erzeugen-x3dom-webgl-344872/, 12.03.2014
- http://www.igd.fraunhofer.de/Institut/Abteilungen/Visual-Computing-System-Technologies/Projekte/X3DOM, 12.03.2014
- http://www.leed.ch/history/x3d/werkzeuge.htm, 12.03.2014
- http://www.leed.ch/history/x3d/zweck.htm, 12.03.2014
- http://www.web3d.org/realtime-3d/x3d/what-x3d, 10.03.2014
- http://x3dom.org/docs/dev/tutorial/blender_export.html, 4.3.2014
- http://x3dom.org/docs/dev/tutorial/dataconversion.html#dataconversion, 4.3.2014
- http://www.x3dom.org/?page_id=2, 4.3.2014
- https://doc.x3dom.org/tutorials/animationInteraction/picking/index.html