Einführung in X3D

Ein kurzes Tutorial

Jan-Martin Kirves

Inhaltsverzeichnis

1. Was ist X3D

Bei X3D handelt es sich um eine Beschreibungssprache für 3D Szenen, die im XML Format gehalten ist. Sie entwickelte sich aus der in den 90er Jahren definierten Virtual Reality Modeling Language (VRML) und ist seit 2004 ein offener ISO-Standard.

Hauptsächlich wird X3D zur Content Beschreibung oder Austausch von 3D Szenen genutzt. Dabei ist das Format Human readabel und kann auch genutzt werden um ohne externe Tools 3D Content „von Hand“ zu erzeugen.

Die Webseite des Web3d Consortiums ist unter:

http://www.web3d.org/realtime-3d/x3d/what-x3d

zu finden und bietet viele nützliche Informationen.

Die DTD des X3D Formats mit hilfreichen Erläuterungen und Kommentaren findet sich unter:

http://www.web3d.org/x3d/content/X3dTooltips.html .

2. Wie betrachtet man X3D Dateien

Um die Inhalte einer X3D Datei im XML Format zu betrachten, reicht ein üblicher Text Editor. Mit diesem lassen sich die Dateien lesen und bearbeiten.

Möchte man jedoch den Inhalt als 3D Szene betrachten, so braucht man einen Interpreter oder Player. Es gibt zwei übliche Player, den freien Instantreality Instant Player vom Fraunhofer Institut und den proprietären Otaga Player, welcher neben der Fähigkeit 3d Inhalte anzuzeigen auch ein Plugin für einige Webbrowser bietet, um 3d Inhalte online darstellen zu können.

Für die Darstellung von X3D Inhalten im Webbrowser ist das opensource Framework x3dom (gesprochen X Freedom) eine ausgezeichnete Möglichkeit.

Eine Webseite, die X3D Inhalte darstellen möchte, muss dafür nur 2 Dateien, eine Javascript- und eine CSS-Datei, einbinden und sollte im xhtml Format geschrieben sein. Dies versetzt die meisten modernen Browser in die Lage X3D Inhalte direkt ohne Plugins anzeigen zu können, die entsprechende Hardware vorausgesetzt.

3. Grundelemente einer 3d Szene

Die vier Grundelemente einer 3d Szene sind die folgenden: (1) Ein dreidimensionaler Raum, in dem alle weiteren Elemente der Szene angeordnet werden können. (2) Objekte, die dargestellt werden sollen. (3) Lichtquellen, um die Objekte zu beleuchten und somit sichtbar zu machen, sowie (4) eine Kamera aus deren Blickwinkel ein Bild gerendert wird.

Im folgenden Bild (Abbildung 1) sind die einzelnen Elemente noch einmal illustriert. Der große transparente Kubus steht für den Raum, die blauen kleinen Kuben für 3d Objekte der Szene. Es werden drei verschiedene Lichtquellen gezeigt, die große Sonne ist eine Richtungs-Lichtquelle, sie beleuchtet alle Objekte gleichmäßig. Die kleine Sonne steht für eine Punkt-Lichtquelle. Bei dieser wird Licht von einem Punkt aus in alle Richtungen emittiert. Der kleine Stern steht für ein Spot-Licht,

welches Objekte in einem definierten Kegel erleuchtet. Der Smiley steht für die Kamera, die die Szene erfasst und der orange umrahmte Bereich für den Blickwinkel der Kamera.

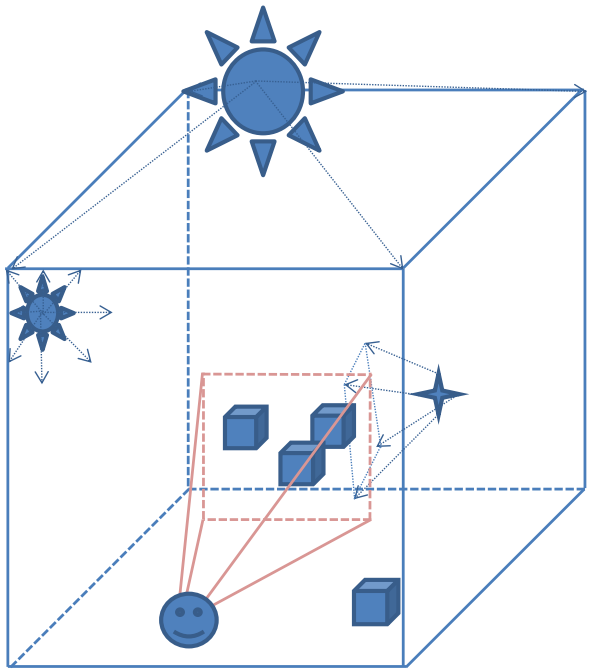

Abbildung 1: 3D Szene

Das nächste Bild (Abbildung 2) zeigt das gerenderte Bild der oben beschriebenen Szene, das durch den Blickwinkel der Kamera definiert wird. Zu sehen ist, dass nur die Objekte innerhalb des Blickwinkels der Kamera dargestellt werden. Die Objekte sind oben am hellsten, da sie dort von der Richtungs-Lichtquelle beleuchtet werden und werfen angedeutete Schatten in Richtung der Punkt- und Spot-Lichtquelle.

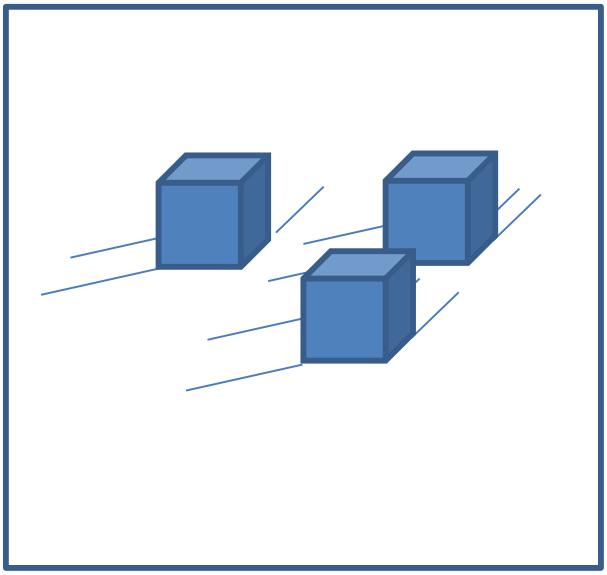

Abbildung 2: Gerenderte Szene

4. Wie nutzt man X3D

Im Folgenden wird erklärt, wie man eine einfache Szene mit Hilfe von X3D erzeugen kann und diese über x3dom im Webbrowser darstellt.

4.1.Vorbereitung

Die Beispiele sind alle mit dem Firefox Webbrowser Version 21 getestet. Zum lokalen Ausführen der Beispiele muss eventuell noch eine Konfiguration am Firefox vorgenommen werden, die es erlaubt Inhalte vom lokalen Dateisystem nachladen zu dürfen.

Dazu muss folgendes getan werden:

1.) In der Adresszeile, Abbildung 3, „***about:config***“ eingeben

Abbildung 3 Firefox Adresszeile, about:config

2.) Die Sicherheitswarnung, Abbildung 4, bestätigen

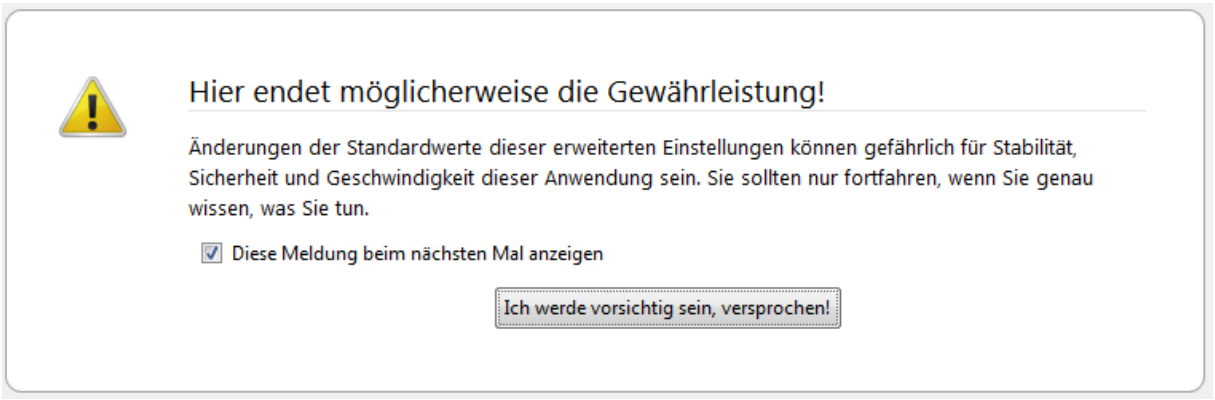

Abbildung 4: Firefox Konfigurations Warnung

3.) Als Suchfilter „***fileuri***“ eingeben (Abbildung 5)

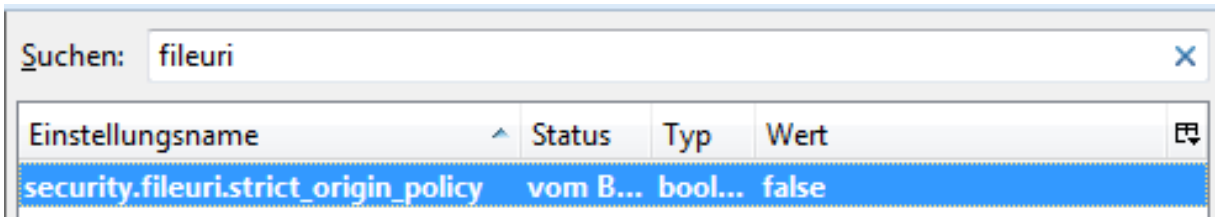

Abbildung 5: Firefox Einstellungs Dialog

4.) Die Option „***security.fileuri.strict_origin_policy***“ auf „***false***“ setzen (Abbildung 5).

Für die Einstellungen für andere Browser oder weitere Informationen ist folgender Link hilfreich:

http://x3dom.org/docs/dev/notes/cors.html

Des Weiteren sollte die Grafikkarte in der Lage sein 3D Inhalte über WebGL darstellen zu können. Eine Liste mit Grafikkarten, die nicht unterstützt werden, findet sich hier:

http://www.khronos.org/webgl/wiki/BlacklistsAndWhitelists

4.2.Die Beispiele öffnen

Um die Beispiele betrachten zu können, muss der komplette Ordner X3D auf das lokale System kopiert werden um dann die Datei „00_show.xhtml“ im Firefox zu öffnen.

In dieser xhtml Datei (Abbildung 6) sind alle Beispiele verlinkt und können im Browser betrachtet werden.

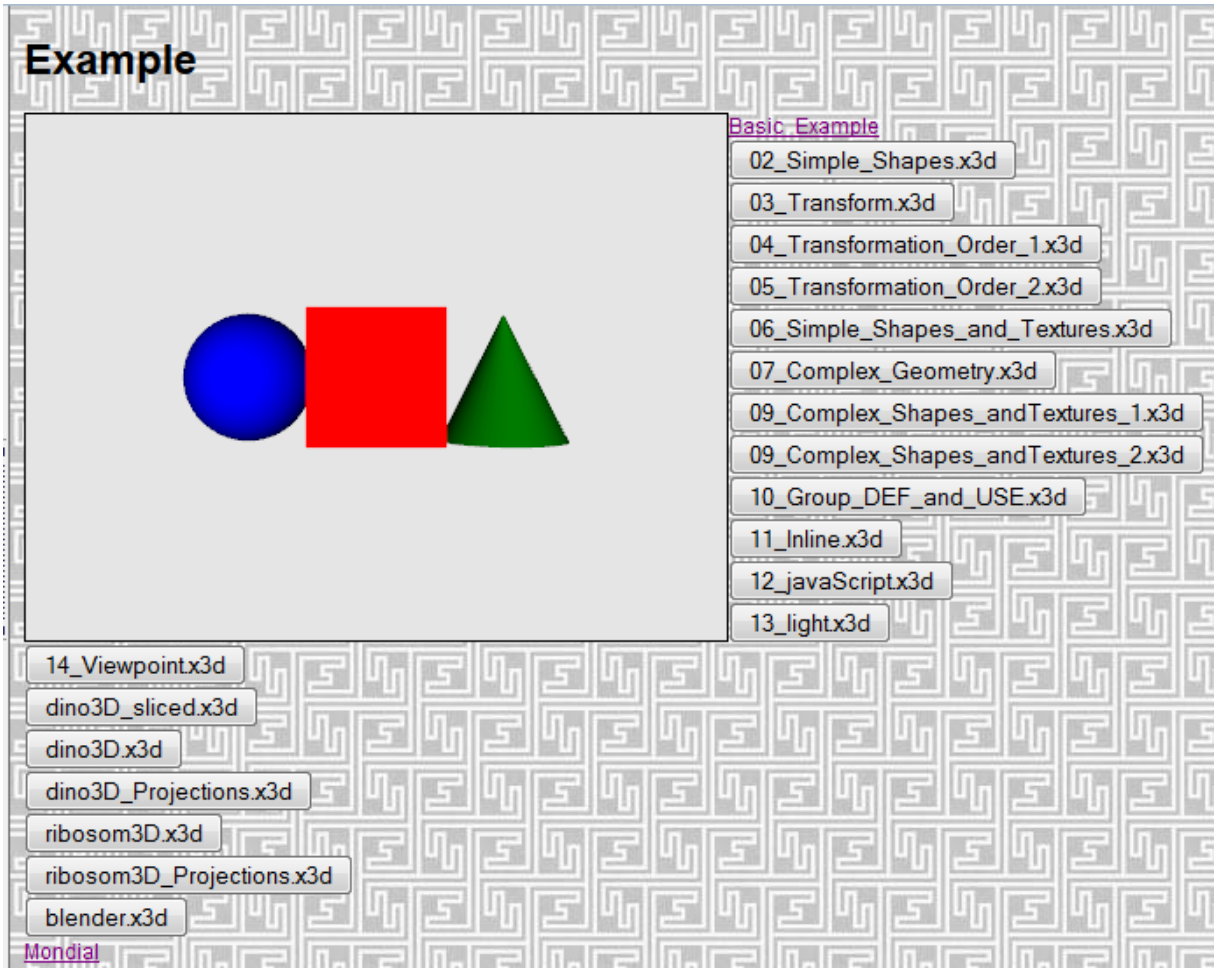

Abbildung 6: Darstellung von 00_show.xhtml als Zugriffspunkt auf alle Beispiele

Die Links „Basic Example“ und „mondial“ führen zu eigenständigen xhtml Dateien, alle anderen Beispiele laden dynamisch X3d Inhalte in das Dokument „00_show.xhtml“.

5. Wie erzeugt man X3D Inhalte

Es gibt mehrere Möglichkeiten X3D Inhalte zu erzeugen. Die erste wäre das Schreiben von X3D Dateien per Hand in einem Text Editor. Die meisten Beispiele sind auf diese Art entstanden.

Eine zweite Möglichkeit wäre die Verwendung eines 3D Modellierprogramm um eine 3D Szene zu erzeugen und diese dann ins X3D Format zu exportieren. Ein solches 3D Modellierprogramm wäre

z.B. „Blender“(Abbildung 7). Es ist sehr mächtig, frei verfügbar und bietet nativen X3D Ex- und Import.

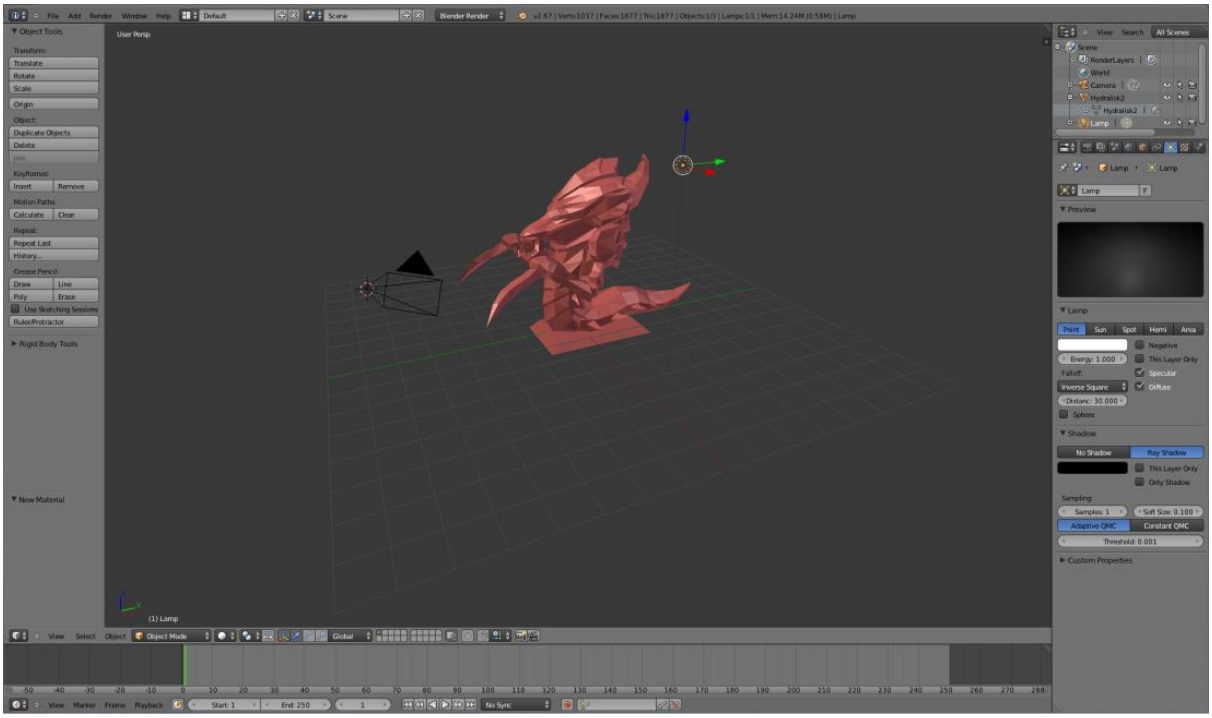

Abbildung 7: Screenshot vom Programm "Blender"

Das Programm steht in Versionen für alle gängigen Betriebssysteme zur Verfügung und kann heruntergeladen werden unter:

http://www.blender.org/

Ein Beispiel für das oben im „Blender“ gezeigte 3D Modell als X3D Export findet sich in Beispiel „blender.x3d“ (Abbildung 8).

Abbildung 8: Screenshot Blender X3D Export (blender.x3d)

Eine dritte Möglichkeit ist das Erzeugen einer X3D Datei aus einer anderen xml Datei heraus mit Hilfe eines XSLT Stylesheets. Diese Methode wird später noch einmal genauer erläutert.

6. Grundaufbau und die Einbindung in ein HTML Dokument

Das folgende Code Segment zeigt den Inhalt einer einfachen HTML Datei, die ein grundlegendes Beispiel für X3D Inhalte enthält. Es handelt sich um das Beispiel „Basic Example“ in der Datei „01_BasicExample.xhtml“ (Abbildung 9).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>X3D Examples</title>
    <link rel="stylesheet" type="text/css" href="x3dom.css" />
    <script type="text/javascript" src="x3dom.js"></script>
  </head>
  <body style="background-color:#E0E0E0; background-image:url(background.png); ">
    <h1>Example</h1>
     <X3D xmlns="http://www.web3d.org/specifications/x3d-namespace"  width="400px" height="300px">
       <scene>
         <Background DEF="bgnd" transparency="0.5" skyColor="0 1 0"/>
         <shape>
           <appearance>
             <material diffuseColor='red'></material>
           </appearance>
           <box></box>
         </shape>
       </scene>
     </X3D>
  </body>
</html>
```

Abbildung 9: Inhalt von 01_basic_example.xhtml

Zu beachten ist, dass eine XHTML Datei verwendet wird, weswegen auch der entsprechende DOCTYPE gesetzt werden muss. Da zwei verschiedene XML Sprachen genutzt werden muss auch entsprechend der namespace gesetzt werden, für den html Bereich, sowie für den X3D Bereich.

Die für x3dom benötigten Dateien sind die x3dom.js Javascript Datei, die über den script Tag im html Header eingebunden wird, sowie das x3dom.css Stylesheet, das über den link Tag im html Header eingebunden wird.

Die folgende Abbildung 10 zeigt die Darstellung der Datei „01_basic_example.xhtml“ im Webbrowser.

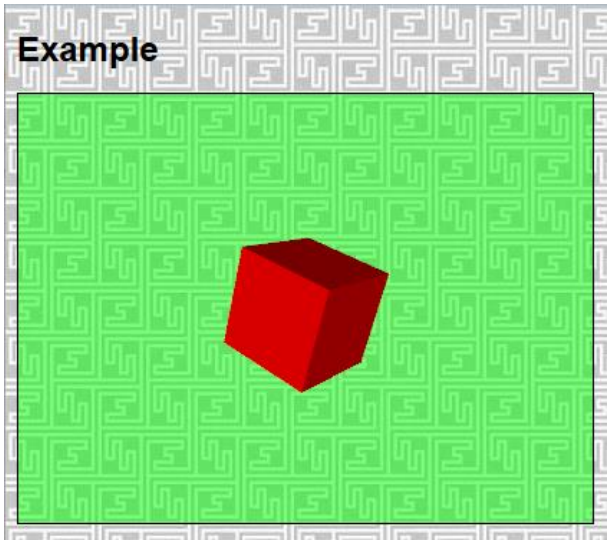

Abbildung 10: Darstellung von 01_basic_example.xhtml. XHTML Seite mit einer 3D Szene mit grün-transparentem Hintergrund und einem rotem Würfel

X3D Inhalte befinden sich immer in einer <X3D> (http://www.web3d.org/x3d/content/X3dTooltips.html#X3D) Umgebung. In dieser befindet sich eine <scene> (http://www.web3d.org/x3d/content/X3dTooltips.html#Scene) Umgebung, welche den Raum, eine Kamera und eine Richtungslichtquelle aus Blickrichtung der Kamera für die 3D Darstellung bereit stellt. Über ein <Backgroud/> (http://www.web3d.org/x3d/content/X3dTooltips.html#Background) Element kann der Hintergrund der Szene definiert werden. Im Beispiel wird über das „skyColor“ Attribut die Farbe auf grün gesetzt und über das „transparency“ Attribut wird der Hintergrund halb durchscheinend.

Dreidimensionale Objekte werden immer in einer <shape> (http://www.web3d.org/x3d/content/X3dTooltips.html#Shape) Umgebung definiert. Diese Umgebung enthält eine < appearance> (http://www.web3d.org/x3d/content/X3dTooltips.html#Appearance) Umgebung, welche für die Erscheinung des Objektes verantwortlich ist. Sie kann eine <material> (http://www.web3d.org/x3d/content/X3dTooltips.html#Material) Umgebung enthalten, über die z.B. die Farbe des Objektes definiert werden kann. Im Beispiel wird durch das Attribut „diffuseColor“ die Farbe auf rot gesetzt. Die <shape> Umgebung enthält des Weiteren eine Geometrie. Diese kann auf verschiedene Weise definiert werden. Im Beispiel wird eine einfache Form verwendet, die von X3D direkt erzeugt wird und zwar über die <box> (http://www.web3d.org/x3d/content/X3dTooltips.html#Box) Umgebung, welche einen Einheitswürfel erzeugt.

Auf diese Weise wird eine Szene mit grün durchscheinendem Hintergrund und einem roten Würfel als 3D Objekt erzeugt, wie sie in der vorherigen Abbildung zu sehen ist.

6.1.Einfache Standardformen

Weitere Formen sind <sphere> (http://www.web3d.org/x3d/content/X3dTooltips.html#Sphere) welche eine Kugel erzeugt, <cone> (http://www.web3d.org/x3d/content/X3dTooltips.html#Cone) welche einen Kegel erzeugt und <cylinder> (http://www.web3d.org/x3d/content/X3dTooltips.html#Cylinder) welche einen Zylinder erzeugt. Alle einfachen Formen lassen sich auch direkt mit einer vorgegeben Größe erzeugen. Details der einfachen Formen finden sich in den

entsprechenden Links. Das Beispiel „02_SimpleShapes.x3d“ zeigt die eben genannten Formen und ist in der folgenden Abbildung 11 zu sehen.

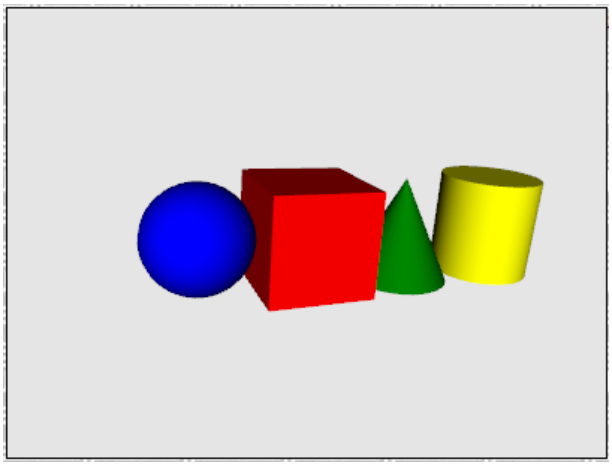

Abbildung 11: Darstellung von 02_Simple_Shapes.x3d. Blaue Kugel, roter würfel, grüner Kegel und gelber Zylinder

7. Objekte Transformieren

Um Objekte im 3D Raum anordnen und ausrichten zu können hält X3D die <transform> (http://www.web3d.org/x3d/content/X3dTooltips.html#Transform) Umgebung bereit. Alle Elemente, die sich in dieser Umgebung befinden, werden den Attributen „translation“, „rotation“ und „scale“ entsprechend transformiert. Es handelt sich dabei stets um eine Änderung des lokalen Koordinatensystems aller Kind-Elemente der <transform> Umgebung.

Das „translation“ Attribut verschiebt alle Kind-Elemente relativ um die gegebenen Werte in Richtung der entsprechenden Achse. Das „rotation“ Attribut rotiert alle Kind-Elemente um eine durch die ersten drei Werte definierte Achse um einen Winkel der durch den vierten Wert im Bogenmaß definiert wird. Das „scale“ Attribut skaliert die Größe aller Kind-Elemente. Dabei wird die Größe der entsprechenden Achse mit dem Wert des „scale“ Attributes multipliziert.

Die Reihenfolge der Transformationsoperationen ist immer dieselbe, ungeachtet der Reihenfolge der Attribute. **Die Reihenfolge ist immer „translation“ -> „rotation“ -> „scale“** . Intern wird dabei eine 4x4 Transformationsmatrix aufgebaut, mit der die 3D Objekte multipliziert werden.

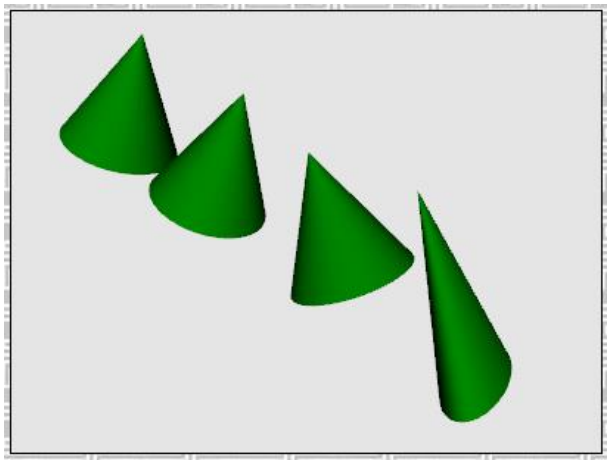

Abbildung 12: Darstellung von 03_Transform.x3d. 1. Verschiebung. 2. Verschiebung und Drehung. 3. Verschiebung, Drehung und Skalierung

Das Beispiel „03_Transform.x3d“ (Abbildung 12) zeigt die Anwendung der <transform> Umgebung, von links nach rechts, am Beispiel eines grünen Kegels, der zuerst nur verschoben wird, als zweites verschoben und um 45° gedreht und als drittes verschoben, um 45° gedreht und in x-Richtung um 0.5 in y-Richtung um 1.5 skaliert wird

7.1.Reihenfolge von Transformationen

Wie zuvor erklärt ist die Reihenfolge von Transformationen einer <transform> Umgebung immer dieselbe, wenn nun eine andere Reihenfolge gewünscht wird müssen mehrere <transform> Umgebungen ineinander geschachtelt werden. Auf diese Weise werden die Transformationen nacheinander aufeinander angewendet. **Die Reihenfolge ist immer Außen -> Innen.** Intern werden die Transfomations-Matrizen von innen nach außen multipliziert, was zu dem Effekt führt, dass die als letztes multiplizierte Matrix zuerst angewendet wird.

Die Reihenfolge von Transformationen wird in den Beispielen „04_Transformation_order_1.x3d“ und „05_Transformation_Order_2.x3d“ illustriert, wobei vier explizite Reihenfolgen verdeutlicht werden.

Beispiel „04_Transformation_order_1.x3d“ zeigt die folgenden zwei Reihenfolgen:

```
<transform translation=" 2 0 0">
 <transform rotation=" 0 0 1 0.785">
  <transform scale="0.5 0.5 0.5">
```

Abbildung 13: Codefragment Verschiebung->Rotation->Skalierung

In Abbildung 13 wird zuerst um 2 Einheiten verschoben und dann um PI/4 = 0.785 (40°) um die z-Achse rotiert, und dann auf jeder Achse mit den Faktor 0.5 verkleinert. Abbildung 15 zeigt dies vom grünen Kegel hin zum roten Kegel.

```
<transform rotation=" 0 0 1 0.785">
 <transform translation=" 2 0 0">
  <transform scale="0.5 0.5 0.5">
```

Abbildung 14: Codefragment Rotation->Verschiebung->Skalierung

In Abbildung 14 wird zuerst um PI/4 = 0.785 (40°) um die z-Achse gedreht, was zu einer Drehung des lokalen Koordinatensystems führt, sodass die folgende Verschiebung um 2 Einheiten in Richtung x-Achse nach rechts oben im 45° Winkel passiert und zum Schluss wird noch einmal um den Faktor 0.5 verkleinert. Die folgende Abbildung 15 illustriert dies vom grünen Kegel in zum blauen Kegel.

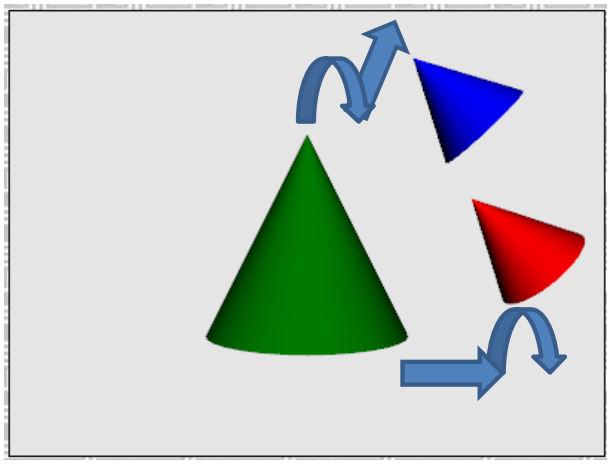

Abbildung 15: Darstellung von 04_Transformation_order_1.x3d. Reihenfolge grün->rot ist Verschiebung, Drehung und Skalierung. Reihenfolge grün->blau ist Drehung, Verschiebung und Skalierung

Beispiel „04_Transformation_order_1.x3d“ zeigt die Reihenfolgen, welche Verschiebung und Skalierung betreffen.

```
<transform translation=" 4 0 0">
        <transform scale="0.5 0.5 0.5">
```

Abbildung 16: Codefragment Verschiebung->Skalierung

In Abbildung 16 wird zuerst um 4 Einheiten in Richtung der x-Achse verschoben und dann um den Faktor 0.5 verkleinert. Dies wird in der folgenden Abbildung 18, vom grünen Kegel hin zum roten Kegel, dargestellt.

```
<transform scale="0.5 0.5 0.5">
  <transform translation=" 4 0 0">
```

Abbildung 17: Codefragment Skalierung->Verschiebung

Bei dieser, in Abbildung 17 gezeigten, Reihenfolge wird zuerst um den Faktor 0.5 verkleinert, was das lokale Koordinatensystem verkleinert, weshalb sich die folgende Verschiebung um 4 Einheiten in Richtung der x-Achse in einer Verschiebung um 2 Einheiten, global betrachtet, auswirkt. Die nachfolgende Abbildung 18 zeigt dies vom grünen Kegel hin zum blauen Kegel auf.

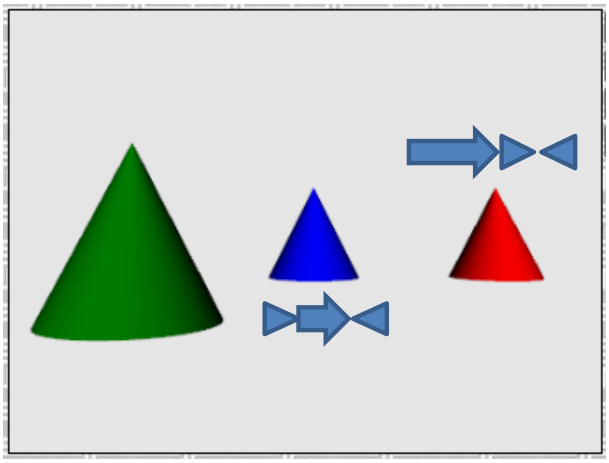

Abbildung 18: Darstellung von 04_Transformation_order_1.x3d . Reihenfolge grün->rot, Verschiebung->Skalierung. Reihenfolge grün->rot Skalierung->Verschiebung

8. Einfache Formen und Texturen

Es ist auch möglich Objekte nicht nur einfarbig darzustellen, sondern ein zweidimensionales Bild auf deren Oberfläche zu legen. Ein solches Bild wird Textur genannt und kann über das <ImageTexture/> (http://www.web3d.org/x3d/content/X3dTooltips.html#ImageTexture) Element definiert werden. Dabei wird das Bild über das „url“ Attribut referenziert und sollte im PNG oder JPG Format vorliegen, da diese Formate von jeder X3D Implementierung unterstützt werden.

Das Beispiel „06_Simple_Shapes_and_Textures.x3d“ zeigt die Anwendung des <ImageTexture/> Elements.

```
<shape>
 <appearance>
  <material diffuseColor='red'></material>
  <ImageTexture url="Earth-640x360.png"/>
 </appearance>
 <box></box>
</shape>
```

Abbildung 19: Codefragment für Textur anwendung

Man sieht hier (Abbildung 19), dass das <ImageTexture/> Element in der <apperance> Umgebung einer <shape> Umgebung angesiedelt ist. Das „url“ Attribut verweist auf das Bild „Earth-640x360.png“ welches in Abbildung 20 zu sehen ist.

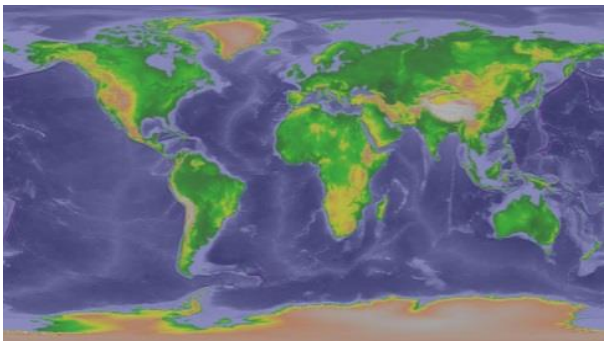

Abbildung 20: Textur Earth-640x360.png

Die Textur wird auf die Geometrie eines Würfels angewendet, wobei X3D bei einfachen Formen selbst darüber entscheidet, wie eine Textur auf ein Objekt gelegt wird. In Abbildung 21 sieht man das Ergebnis, wobei zu beachten ist, dass die Textur auf die Vorderseite des Würfels skaliert wurde und dann um den Äquator herum wiederholt wurde. Ober- und Unterseite wurden von der Vorderseite aus wiederholt angewendet.

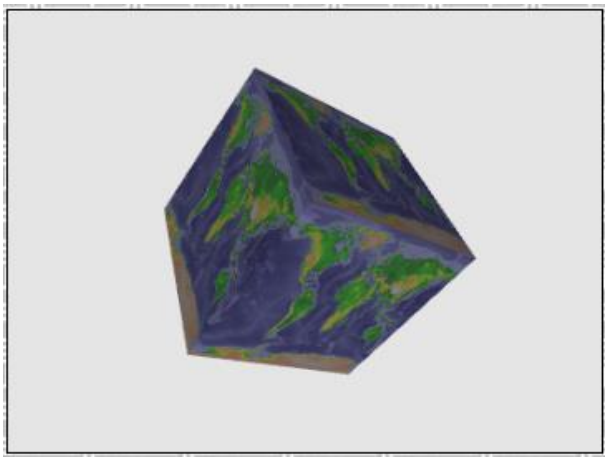

Abbildung 21: Würfel mit Textur

Wird im Code aus Abbildung 19 die <box> Umgebung durch einen <sphere> Umgebung ersetzt, so erhält man, wie in Abbildung 22 zu sehen, eine Kugel, die die Erde darstellt.

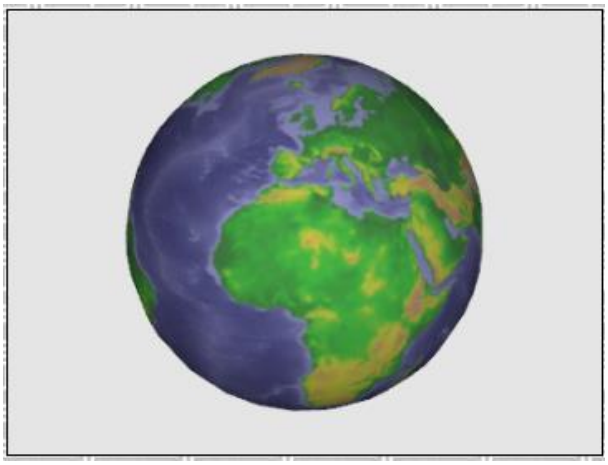

Abbildung 22: Kugel mit Textur

Man kann sehen, dass die Textur aus Abbildung 20 eine auf ein Rechteck verzerrte Kugeloberfläche ist. Durch das Aufspannen auf einen dreidimensional Kugel wird die Verzerrung wieder rückgängig gemacht, was zur Darstellung eines Globus führt.

9. Komplexe Geometrie

Die bisher vorgestellten geometrischen Objekte wurden direkt von X3D erzeugt und boten nur wenig Spielraum für Anpassungen. Eine 3D Szene könnte mit ihrer Hilfe nur wie mit Bauklötzen gebaut erstellt werden.

Möchte man nun aber frei dreidimensionale Objekte erstellen, ähnlich als würde man eine Skulptur modellieren, benötigt man eine weitere Umgebung, und zwar die <IndexedFaceSet> (http://www.web3d.org/x3d/content/X3dTooltips.html#IndexedFaceSet) Umgebung.

Ein beliebiges 3D Objekt besteht immer aus einer Anzahl von Punkten im Raum. Über diese Punkte wird ein Polygonnetz gespannt. Die einzelnen Polygone dieses Netzes bilden dann die Flächen des 3D Objektes, die dargestellt werden. Um nun eine Punktwolke für ein 3D Objekt zu erzeugen, benötigt man das <Coordinate/> (http://www.web3d.org/x3d/content/X3dTooltips.html#Coordinate) Element, dass ein Kind der <IndexedFaceSet> Umgebung ist. In diesem Element werden die Koordinaten der Punkte als aufeinander folgende Tripel von Werten im „point“ Attribut angegeben, in Abbildung 23 ist dies rot markiert zu sehen.

```
<IndexedFaceSet DEF='haus-idx' coordIndex='0 2 1 -1 2 0 3 -1 10 5 11 -1 5 10 4 -1 4 6 5 -1 6 4 7 -1 7 2
6 -1 2 7 1 -1 2 3 8 -1 11 9 12 -1 9 11 5 -1 5 6 9 -1 6 8 9 -1 8 6 2 -1 0 1 4 -1 1 7 4 -1'>
        <Coordinate DEF='haus-coordR' point='0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 0 1 1 1 1 1 1 0 1 0.5 1.5 0
0.5 1.5 1 0 0 0 0 1 0 0.5 1.5 0'/>
 </IndexedFaceSet>
```

Abbildung 23: Codefragment IndexedFaceSet

Die Koordinaten des <Coordinate> Elements sind noch einmal in Abbildung 24 als rote Kreise Dargestellt.

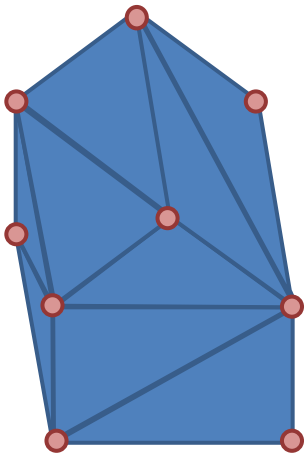

Abbildung 24: Visualisierung eines IndexedFaceSets mit Coordinate Punkten

Um nun ein Polygonnetz über diesen Punkten aufzuspannen, wird nun in der das <Coordinate/> Element beherbergenden <IndexedFaceSet> Umgebung das „coordIndex“ Attribut genutzt. Dabei werden die Indices, der Punkte, die ein Polygon bilden, hintereinander angegeben. Verschiedene Polygone werden dabei durch „-1“ getrennt. In Abbildung 23 ist die genaue Notation im blau markierten Text für Dreiecke zu sehen. Die Flächen des Polygonnetz sind dann noch einmal als blaue Dreieicke in Abbildung 24 veranschaulicht.

Die eben beschriebene komplexere geometrische Form ist in der Beispiel Datei „07_Complex_Geometry.x3d“ enthalten. Abbildung 25 zeigt wie das Polygon Netzt gerendert aussieht.

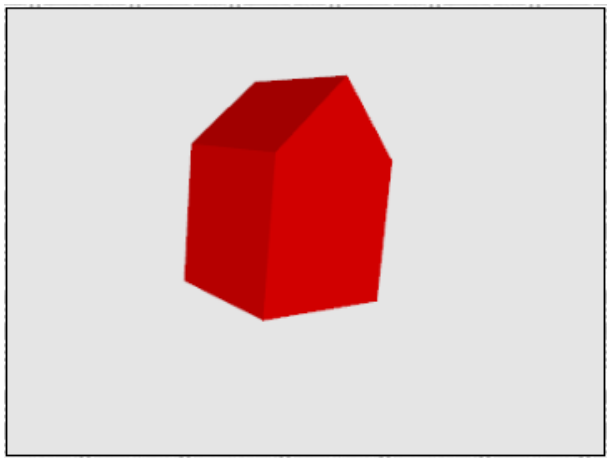

Abbildung 25: Darstellung von 07_Complex_Geometry.x3d

10. Komplexe Geometrie und Texturen

Wenn man nun eine Textur auf ein komplexes geometrisches Objekt legen möchte, ist ein aufwändigeres Vorgehen, als das für die einfachen Standarformen beschrieben, nötig. Versucht man z.B. die Textur aus Abbildung 26 wie zuvor beschrieben nur mit dem <ImageTexture/> Element auf das in Abbildung 23 beschrieben Objekt anzuwenden, erhält man leider nicht ein dreidimensionales Haus mit Fenstern, Türen Dach und Giebel an den korrekten Positionen. Stattdessen erhält man ein verzerrtes Abbild wie der Beispieldatei „09_Complex_Shapes_andTextures_1.x3d“ gezeigt und in Abbildung 27 zu sehen ist.

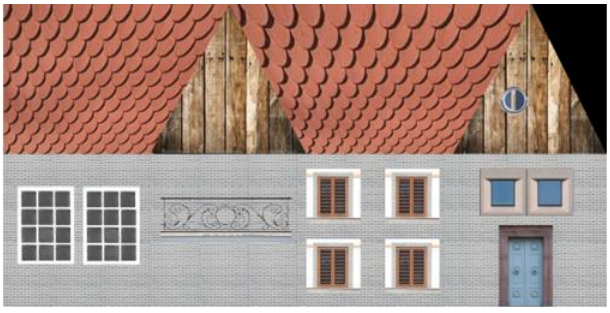

Abbildung 26: Verzerrte Haus Textur für komplexe geometrische Objekte

Das Problem liegt darin, dass X3D nicht in der Lage ist, die Textur auf die gegebene Punktwolke in gewünschter Art und Weise anzuwenden. Die Textur wird einfach auf den kleinstmöglichen, das Objekt umgebenden Quader angewendet und dann interpoliert.

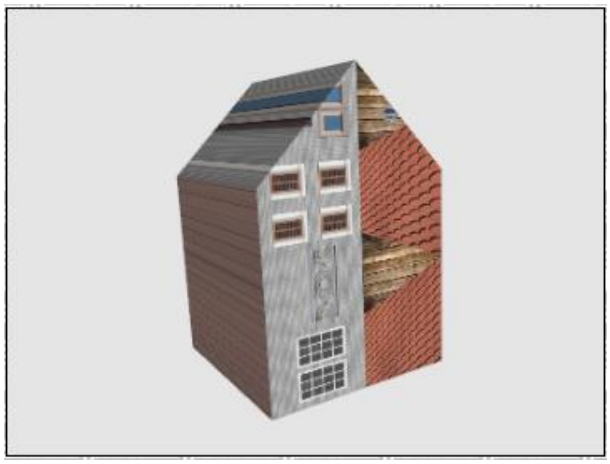

Abbildung 27: Darstelung von 09_Complex_Shapes_andTextures_1.x3d

Um die Textur wie gewünscht auf das dreidimensionale Objekt zu legen, ist ein Texture Mapping notwendig. Dabei wird jedem Punkt des Objektes, der im <Coordinate/> Element definiert ist, ein Punkt aus der Textur, die im <ImageTeexture/> Element definiert ist, zugewiesen. Die Punkte der Textur werden über das <TextureCoordinate/> (http://www.web3d.org/x3d/content/X3dTooltips.html#TextureCoordinate) Elemente definiert. Dort wird im „point“ Attribut eine Liste von Paaren angegeben, wobei jedes Paar von Zahlen für einen Punkt auf der Textur steht. Die Texturkoordinaten werden immer zwischen 0;0 und 1;1 angegeben, dabei ist 0;0 links unten auf der Textur und 1;1 rechts oben. Standartmäßig wiederholt sich eine Textur zyklisch, sodass Texturkoordinaten größer 1 oder kleiner 0 sich in derselben Textur wieder finden. Die Koordinaten 0.5;0.5 und 1.5;1.5 würden den selben Punkt der Textur, nämlich die Mitte, bezeichnen.

Die Zuordnung von Texturkoordinaten zu den Punkten des 3D Modells definiert sich über die Reihenfolge in der beide angegeben werden. Dem ersten Punkt des 3D Modells wird die erste Texturkoordinate zugeordnet.

In Abbildung 28 ist das Codefragment der Texturkoordinaten gezeigt, welches die Textur aus Abbildung 26 auf das geometrische Objekt aus Abbildung 23 legt:

```
<TextureCoordinate DEF='haus-TEXCOORD' point='1 0 0.75 0 0.75 0.5 1 0.5 0.25 0 0.25 0.5 0.5 0.5 0.5
0 0.875 1 0.375 1 0 0 0 0.5 0 1'/>
```

Abbildung 28: Codefragment Texturkoordinaten

Für eine bessere Veranschaulichung welcher Punkt welche Texturkoordinate zugewiesen bekommt, zeigt Abbildung 29 schematisch das zuvor beschriebene 3d Objekt, wobei über jedem Punkt dessen Koordinaten, Index und Textur Koordinaten zu sehen sind.

Aus Sicht der Textur, um zu verstehen, wo dort die Zuordnungen stattfinden, ist in Abbildung 30 die Textur aus Abbildung 26 schematisch dargestellt und für jeden Punkt angegeben, wie seine

Texturkoordinate lautet, so wie die Koordinate das zugehörigen Punktes des 3d Modells und dessen Index.

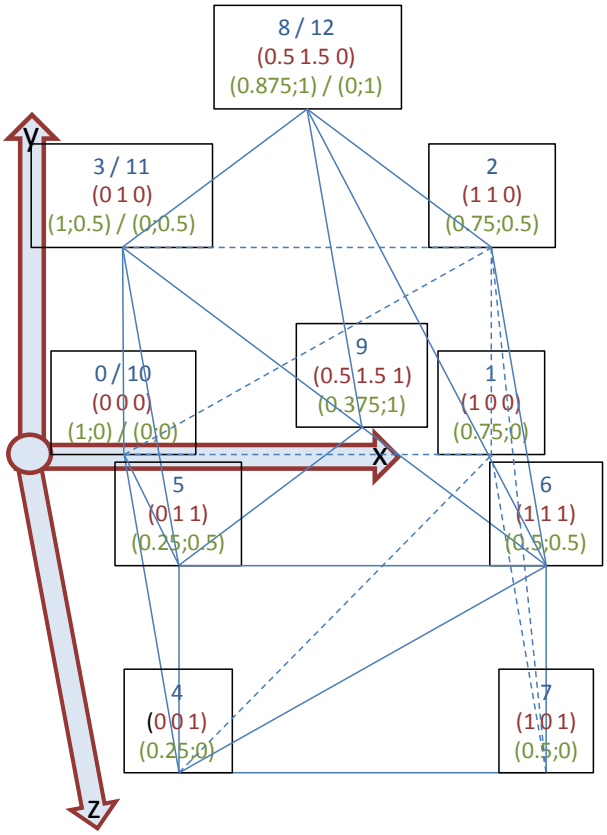

Abbildung 29: Geometrisches Objekt mit Punktkoordinaten, Punktindices und Texturkoordinaten

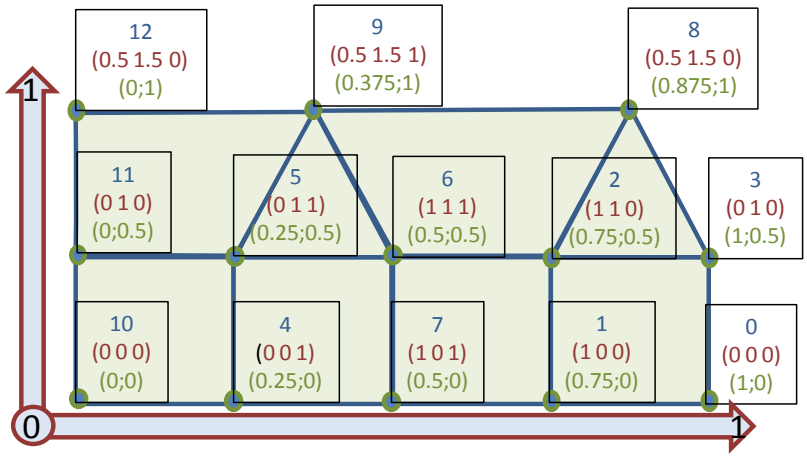

Abbildung 30: Schema für Texturkoordinaten, mit Mapping der Punkt Indices und Punkt Koordinaten

Die Beispiel Datei „09_Complex_Shapes_andTextures_2.x3d“ enthält das oben beschriebene 3d Objekt mit korrekt zugeordneten Texturkoordinaten. In Abbildung 31 ist das Beispiel gezeigt. Zu sehen ist, dass man das geometrische Objekt des Hauses mit korrekten Fenstern, Türen Dach und Giebel dargestellt wird.

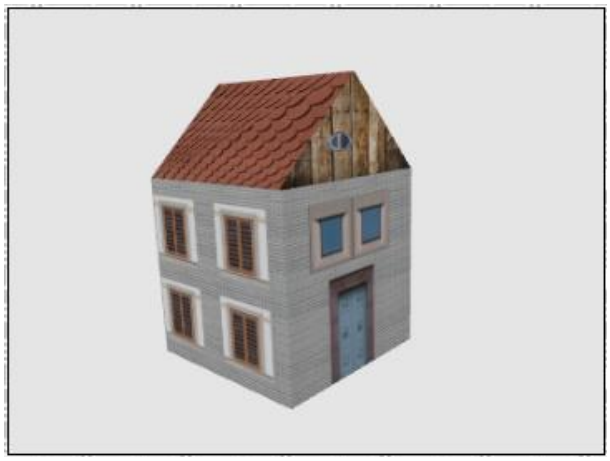

Abbildung 31: Darstellung von 09_Complex_Shapes_andTextures_2.x3d

11. Gruppen und Referenzen

Es ist möglich verschiedene Elemente zu einer Gruppe zusammenzufassen. Dazu wird die <group> (http://www.web3d.org/x3d/content/X3dTooltips.html#Group) Umgebung verwendet. Alle Kind-Elemente der <group> Umgebung werden dabei zu einer Gruppe zusammengefasst.

Jedem Element in X3D kann über das „DEF“ Attribut eine eindeutige ID zugewiesen werden. Über das „USE“ Attribut, welches ebenfalls jedem X3D Element zur Verfügung steht, kann ein über „DEF“ definiertes Element mit allen Kindern wiederverwendet werden. Das Wiederverwenden von vordefinierten Elementen kann die Performance verbessern.

In Beispiel „10_Group_DEF_and_USE.x3d“ ist die <Group> Umgebung genutzt um einen grünen Kegel und einen roten Würfel zu einer Gruppe zusammen zu fassen. Dies ist in Abbildung 32 durch das blaue Oval markiert. Die Gruppe hat über das „DEF“ Attribut die ID „g1“ bekommen.

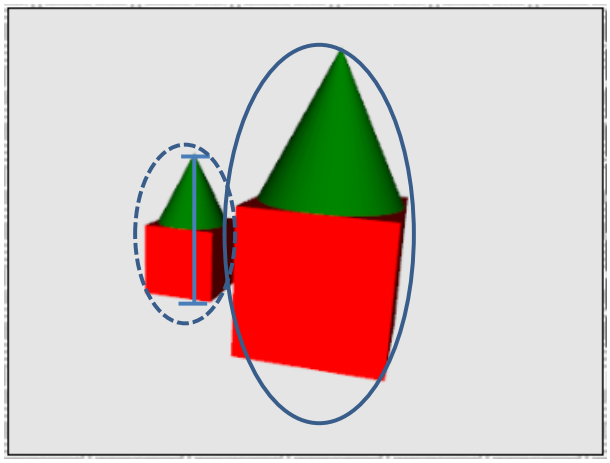

Abbildung 32: Darstellung von 10_Group_DEF_and_USE.x3d mit Markierung für Gruppierung und Transformation

Dann wurde einem neuen <Shape/> Element mit Hilfe des „USE“ Attributes die Gruppe „g1“ zugewiesen, dies ist in Abbildung 32 durch das blaue gestrichelte Oval gekennzeichnet. Diese referenzierte Gruppe wurde nun mit einer <Transform> Umgebung verschoben und verkleinert, was in Abbildung 32 durch die blaue senkrechte Strecke angezeigt werden soll.

12. Import externer Modelle

In einem X3D Dokument müssen sich nicht immer alle Objekte befinden. Es besteht die Möglichkeit ein externes X3D Dokument in ein anderes Dokument einzubinden. Dafür wird das <inline/> (http://www.web3d.org/x3d/content/X3dTooltips.html#Inline) Element verwendet.

```
<X3D xmlns="http://www.web3d.org/specifications/x3d-namespace"  width="400px" height="300px">
 <Scene>
  <Background DEF="bgnd" transparency="0.0" skyColor="0.9 0.9 0.9"/>
  <transform translation=" -2 0 0">
   <Inline url="07_Complex_Geometry.x3d" />
  </transform>
  <transform translation=" 0 0 0">
   <Inline url="09_Complex_Shapes_andTextures_1.x3d" />
  </transform>
  <transform translation=" 2 0 0">
   <Inline url="09_Complex_Shapes_andTextures_2.x3d" />
  </transform>
 </Scene>
</X3D>
```

Abbildung 33: Codefragment zur Nutzung des inline Elements

Das „url“ Attribut des <inline/> Elements gibt die Adresse des zu ladenden X3D Dokuments an. In Beispiel „11_Inline.x3d“ werden die Dokumente „07_Complex_Geometry.x3d“, „09_Complex_Shapes_andTextures_1.x3d“ und „09_Complex_Shapes_andTextures_2.x3d“ über das <inline/> Element in dieselbe Szene geladen und mit der <transform> Umgebungen nebeneinander dargestellt. Die genaue Syntax ist in Abbildung 33 zu lesen. Die Darstellung von Beispiel „11_Inline.x3d“ ist in Abbildung 34 zu sehen.

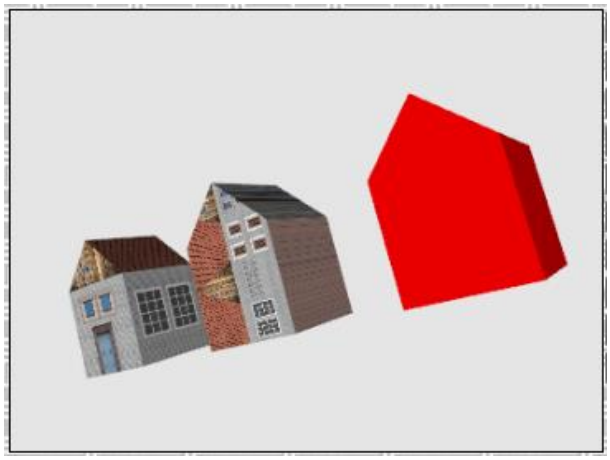

Abbildung 34: Darstellung von 11_Inline.x3d

13. Beleuchtung

Standardmäßig wird eine X3D Szene mit einer „Headlight“ Lichtquelle beleuchtet. Das bedeutet ein gleichmäßiges Richtungslicht, das in Richtung des Kamera-Blickwinkels strahlt, erleuchtet die gesamte Szene einheitlich.

Um nun andere Lichtquellen besser sehen zu können muss das „headlight“ ausgeschaltet werden. Dies geschieht über das als direktes Kind in der <scene> Umgebung angesiedelte <NavigationInfo/> (http://www.web3d.org/x3d/content/X3dTooltips.html#NavigationInfo) Element. In diesem Element muss das Attribut „headlight“ auf „false“ gesetzt werden.

Um nun eine Punkt- und eine Spotlichtquelle einzufügen, wie sie in Abbildung 35 zusehen sind, muss in die <scene> Umgebung ein <PointLight/> (http://www.web3d.org/x3d/content/X3dTooltips.html#PointLight) Element und ein <SpotLight/> (http://www.web3d.org/x3d/content/X3dTooltips.html#SpotLight) Element eingefügt werden. Beispiel „13_light.x3d“ enthält die genaue Syntax.

Eine Lichtquelle hat immer ein „location“ Attribut, welches die Position der Lichtquelle im Raum angibt. Außerdem hat sie ein „radius“ Attribut, welches angibt wie weit die Lichtquelle scheinen soll. Desweitern besitzt sie ein „color“ Attribut, welches die Farbe des Lichtes angibt. Die Farbe eines Objektes ergibt sich dann aus der Multiplikation der Lichtfarbe und der Objektfarbe.

Wenn die Objekte, die von der Lichtquelle angestrahlt werden, einen Schatten werfen sollen, so muss noch das „shadowIntensity“ Attribut der entsprechenden Lichtquelle gesetzt werden. Der Wert gibt an um welchen Faktor die Lichtintensität im Schatten reduziert ist. Zu beachten ist, dass eine Lichtquelle immer alle Objekte bescheint. Ein Lichtstrahl wird also nicht von einem Objekt unterbrochen. Um nur bestimmte Objekte zu beleuchten kann ein Lichtquellen Element tiefer im X3D Baum angesiedelt werden und würde dann nur alle Geschwister sowie deren Kinder beleuchten, wenn das „global“ Attribut der Lichtquelle auf „false“ gesetzt wird.

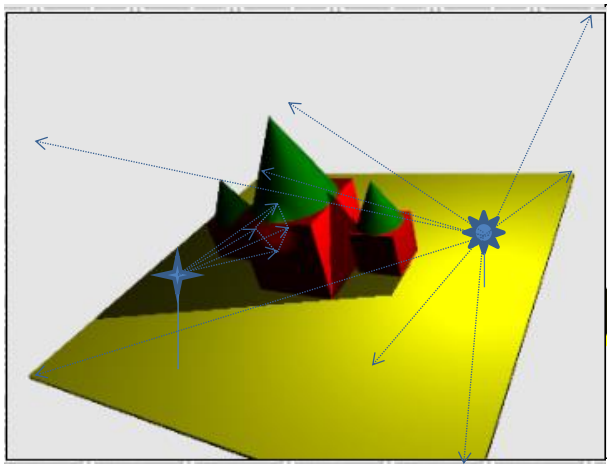

Abbildung 35: Darstellung von 13_light.x3d mit schematischen Lichtquellen. Die Sonne stellt eine Punktlichtquelle dar, der Stern eine Spotlichtquelle, die gepunkteten Pfeile symbolisieren Lichtstrahlen, die von den Quellen ausgehen

Die Spotlichtquelle, in Abbildung 35 als Stern dargestellt, hat neben den oben genannten Attributen noch eine Richtung, welche über das „direction“ Attribut definiert wird und eine Lichtkegelbreite über das „beamWidth“ Attribut definiert wird, sowie eine Angabe wie sich das Licht zum Rand des Kegels hin sich abschwächen soll, über das „cutOffAngle“ Attribut definiert.

14. Blickrichtungen

Die Blickrichtung der Kamera kann jederzeit mit der Maus verändert werden. Es besteht aber auch die Möglichkeit feste, vordefinierte Blickwinkel festzulegen. Diese vorher festgelegten Blickwinkel

können mit den „Bild auf“ und Bild ab“ Tasten in Beispiel „14_Viewpoint.x3d“ durchgegangen werden.

Um eine Blickrichtung vorzudefinieren, ist das <Viewpoint/> (http://www.web3d.org/x3d/content/X3dTooltips.html#Viewpoint) Element nötig. Es beinhaltet unter anderem drei wichtige Attribute. Zum ersten das „position“ Attribut, welches die Position der Kamera im Raum festlegt, als zweites das „fieldOfView“ Attribut, welches die Weite des Blickfeldes angibt und als drittes das „orientation“ Attribut, welches die Blickrichtung der Kamera festlegt.

```
<Viewpoint DEF='str'   position="0 0 5"  fieldOfView='0.76' orientation=" 0  0 1 0"/>
<Viewpoint DEF='u22'   position="0 0 5"  fieldOfView='0.76' orientation=" 1  0 0 0.38"/>
<Viewpoint DEF='l22'   position="0 0 5"  fieldOfView='0.76' orientation=" 0  1 0 0.38"/>
<Viewpoint DEF='d22'   position="0 0 5"  fieldOfView='0.76' orientation="-1  0 0 0.38"/>
<Viewpoint DEF='r22'   position="0 0 5"  fieldOfView='0.76' orientation=" 0 -1 0 0.38"/>
<Viewpoint DEF='fov22' position="0 0 5"  fieldOfView='0.38' orientation=" 0  0 1 0"/>
<Viewpoint DEF='fov90' position="0 0 5"  fieldOfView='1.56' orientation=" 0  0 1 0"/>
<Viewpoint DEF='pos2'  position="0 0 2"  fieldOfView='0.76' orientation=" 0  0 1 0"/>
<Viewpoint DEF='pos20' position="0 0 20" fieldOfView='0.76' orientation=" 0  0 1 0"/>
```

Abbildung 36: Codefragment Blickrichtungen

In Abbildung 36 ist die Syntax für neun verschiedene Blickwinkel angegeben. Der Reihe nach, gerade aus, 45° nach oben, 45° nach links, 45° nach unten, 45° nach rechts. Außerdem ein verengtes Blickfeld, was einem Kamera Zoom entspricht und ein weites Blickfeld, was einem Weitwinkel entspricht. Desweiteren eine nähere Kameraposition und als letztes eine weiter entfernte Kameraposition.

Eine schematische Darstellung der eben genannten Blickwinkel ist in Abbildung 37 zu sehen. Die Reihenfolge von oben ist die gleiche wie von Teilbild A bis I.

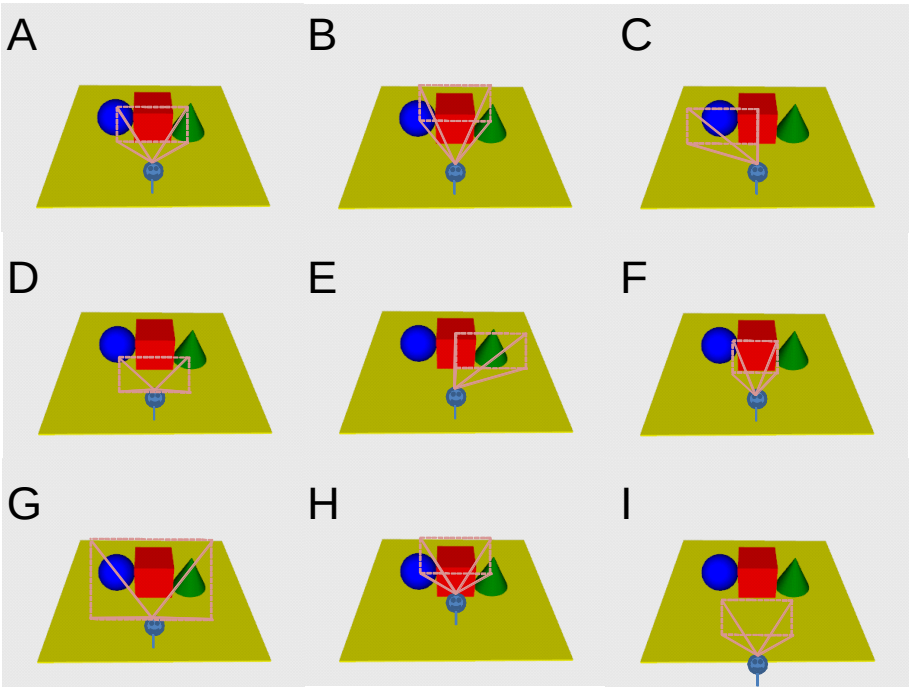

Abbildung 37: Darstellung verschiedener Blickwinkel. A) geradeaus, B) 45° nach oben, C) 45° nach links, D) 45° nach untern, E) 45° nach rechts, F) geradeaus schmales Blickfeld, G) geradeaus weites Blickfeld, H) geradeaus näher dran, I) geradeaus weiter weg

15. JavaScript und x3dom

Um eine, in ein HTML Dokument eingebundene, X3D Szene mit diesem HTML Dokument funktional zu verbinden, bietet x3dom die Möglichkeit an, einer X3D <shape> Umgebung das typische JavaScript „onclick“ Attribut zu setzen. Das hat den Effekt, dass sobald auf ein <shape> Objekt mit gesetztem „onclick“ Attribut geklickt wird, die im Attribut gesetzte JavaScript Funktion eventbasiert aufgerufen wird.

Beispiel „12_javaScript.x3d“ veranschaulicht dies. Hier wurde bei einer Kugel das „onclick“ Attribut mit der Funktion, eine Benachrichtigung auftauchen zu lassen, gesetzt. Diese Benachrichtigung teilt mit, dass die Kugel angeklickt wurde. Bei einem Würfel wurde das „onclick“ Attribut, ähnlich dem der Kugel. Nur, dass der Inhalt der Benachrichtigung anzeigt, dass der Würfel angeklickt wurde.

Bei der Verwendung des „onclick“ Attributes ist zu beachten, dass es nur auf <shape> Objekten verwendet werden sollte. Und es sollte nicht mit Referenzen, die über das „USE“ Attribut referenziert wurden, genutzt werden, da dabei nicht die einzelne Referenz aufgelöst wird, sondern alle Referenzen gleichzeitig.

16. Mit XSLT zu X3D

Da es sich bei X3D um eine xml Sprache handelt, ist eine Transformation von anderen xml Dokumenten hin zu X3D mittels eines XSLT Stylesheets möglich. Dazu muss nur in der zu transformierenden xml Datei nach der „!DOCTYPE“ Definition der Verweis auf das zu nutzende Stylesheet erfolgen. Ein solcher Verweis befindet sich in der Beispiel Datei „mondial.xml“ und ist in Abbildung 38 zu sehen.

```
<?xml-stylesheet type="text/xsl" href="to3d.xsl" ?>
```

Abbildung 38: Codefragment Stylesheet Verweis

Im eigentlichen Stylesheet ist zu beachten, dass Namespaces genutzt werden müssen, da man sich in verschiedenen xml Sprachen bewegt. Dazu kommt, dass sich X3D im default Namespace befinden muss. Die Beispiel Datei „to3d.xsl“ enthält ein solches Stylesheet, mit dem ein Teil des Inhalts der Datei „mondial.xml“ visualisiert werden kann. Eine genauere Beschreibung des Beispiels folgt im späteren Kapitel 18 „Mondial X3D“. Für ein generelles XSLT Stylesheet ist eine bestimmte Präambel wichtig, wie sie in Abbildung 39 beschrieben ist.

```
<xsl:stylesheet
   version="1.1"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:h="http://www.w3.org/1999/xhtml"
   xmlns:d="http://www.web3d.org/specifications/x3d-namespace"
   xmlns="http://www.web3d.org/specifications/x3d-namespace"
   exclude-result-prefixes="html"
>
<xsl:output
   media-type="application/xhtml+xml"
   method="xml"
   indent="yes"
   doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"
   doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"/>
```

Abbildung 39: Codefragment XSLT Präambel

Dabei gibt das erste Element an, dass es sich um ein XSLT Stylesheet handelt und dass die Version 1.1 verwendet wird. Darüber hinaus werden alle nötigen Namespaces definiert. Wie oben schon gesagt ist wichtig darauf zu achten, dass X3D im default Namespace ist, was durch das vorletzte Attribut geschieht. Das zweite Element definiert, wie die Ausgabe der Transformation aussehen soll, das erste Attribut legt fest, dass es sich um das xhtml Format handeln soll, das zweite, dass es sich um ein xml Dokument handelt. Das dritte Attribut sorgt durch eine Einrückung der Ausgabe für bessere Lesbarkeit, die beiden letzten Attribute definieren wo die DTD für xhtml zu finden ist und dass diese anzuwenden ist.

17. X3D und xPath

Auf X3D Daten, die im Browser mittels x3dom dargestellt werden, kann mittels xPath eine Anfrage gestellt werden. Dazu kann die XPath Evaluierung des Browser genutzt werden, die über JavaScript angesprochen wird. Die Evaluierung läuft dabei über das gesamte xhtml Dokument, dessen Teil auch die X3D Daten sind.

Um eine XPath Anfrage stellen zu können muss eine kurze JavaScript Funktion implementiert werden. Der generelle Grundaufbau ist in Abbildung 40 zu sehen.

```
<h:script type="text/javascript">
  <![CDATA[
    function resolver(prefix){
      switch(prefix){
        case “h": return "http://www.w3.org/1999/xhtml";
        case "x": return "http://www.web3d.org/specifications/x3d-namespace";
        default: return "http://www.web3d.org/specifications/x3d-namespace";
      }
    }
    function xpath( exp ){
      var evaluator = new XPathEvaluator();
      var resultxp = evaluator.evaluate(exp, document.documentElement,  resolver, 7, null);
      for(var i=0; i<resultxp.snapshotLength;  i++){
        var currentNodeValue = resultxp.snapshotItem(i);
        //DO SOMETHING….
       }
    }
  ]]>
</h:script>
```

Abbildung 40: Codefragment JavaScript xPath Evaluierung

Die Funktion, die die Evaluierung übernimmt heißt „xpath(exp)“. Dabei ist der Paramenter „exp“ der XPath Ausdruck als String. In der ersten Zeile der Funktion wird ein Evaluierer erzeugt und der Variablen „evaluator“ zugewiesen. In der zweiten Zeile wird der Evaluierer genutzt um die Anfrage aus „exp“ auszuwerten. Dabei ist der erste Parameter die Anfrage, der zweite ist der Einstiegspunkt in das auszuwertende Dokument, der dritte eine Referenz auf einen Namespace Auflöser, der vierte Parameter gibt die Art des zurückgegebenen Resultates an, wobei 7 für ein „Nodeset“ steht und der letzte Parameter kann als Referenz auf ein bestehendes Result Objekt für die Rückgabe genutzt werden. Das Resultat der Auswertung wird der Variablen „xpResult“ zugewiesen. Aus diesem Objekt kann jeder Knotenwert mit der Funktion „snapshotItem(index)“ angesprochen werden. Deshalb wird in einer Schleife über alle Knoten des Result Objektes iteriert, jeder Knoten ausgelesen und entsprechen behandelt.

Der eben erwähnte Namespace Resolver muss auch selbst implementiert werden. Dies geschieht in der Funktion „resolver(prefix)“. Dabei enthält der Parameter „prefix“ die Abkürzung des aufzulösenden Namespace als String. Die Funktion selber enthält nur eine „switch case“ Anweisung, die den vollständigen Namespace der gegeben Abkürzung entsprechend zurück gibt.

Ein detailliertes Beispiel findet sich in der Datei „to3d.xsl“ und wird im Kapitel 18 „Mondial X3D“ noch einmal genauer aufgearbeitet.

18. Mondial X3D

Das Mondial Beispiel verbindet alle beschriebenen Techniken. Es enthält einfach- und komplexe Formen, sowie Texturen für beide Formen. Darüber hinaus werden JavaScript onclick Attribute und Events genutzt und die X3D Szene kann mittels einer XPath Evaluierung modifiziert werden. Das ganze Dokument wird mittels eines XLST Stylesheets erzeugt.

Der Inhalt der Datei „Mondial XML to X3D.xhtml“ wurde mittels des XSLT Stylsheets „to3d.xsl“ aus der Datei „mondial.xml“ erzeugt und ist in Abbildung 41 ausschnittweise dargestellt. Zu sehen ist eine stilisierte Erdkugel, auf der sich kleine Häuser befinden. Diese Häuser stehen jeweils für eine Stadt aus der Datei „mondial.xml“, welche Informationen zu Längen- und Brietengrad zur Verfügung stellt. Auf jedes Haus kann gekickt werden um den Namen der Stadt mittels JavaScript einzublenden. Oberhalb der X3D Szene befindet sich ein Textfeld, welches einen xPath Ausdruck entgegen nimmt um Elemente der X3D Szene auszuwählen. Die gewählten Elemente werden dann aus der Darstellung entfernt und bei erneuter Auswahl wieder eingeblendet. Der Vorgang der Auswahl wird durch den „evaluate xPath“ Button gestartet.

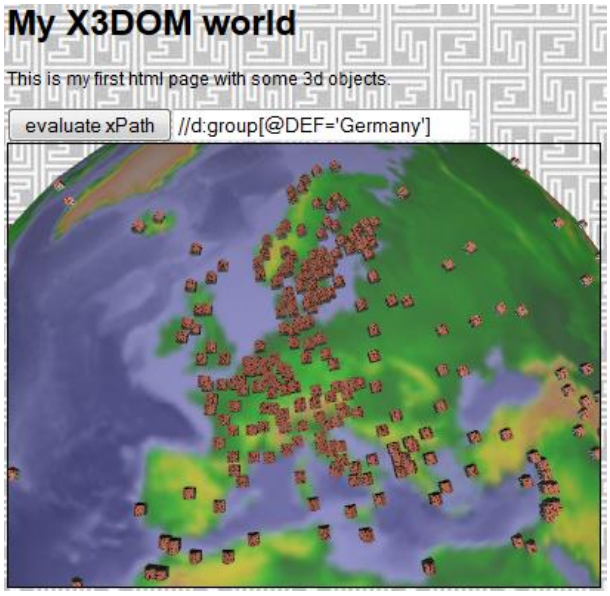

Abbildung 41: Darstellung von Mondial XML to X3D.xhtml

Der Aufruf des XSLT Stylesheets „to3d.xls“ wurde schon in Abbildung 38 aus dem Kapitel 16 „Mit XSLT zu X3D“ gezeigt, im Folgenden wird detailliert auf den Inhalt der Datei „to3d.xsl“ eingegangen.

Die Stylesheet ***Präambel*** ist in Abbildung 39: Codefragment XSLT Präambel zu sehen und wurde in Kapitel 16 „Mit XSLT zu X3D“ erklärt. Der ***xhtml header*** entspricht dem aus Kapitel 6 „Grundaufbau und die Einbindung in ein HTML Dokument“ Abbildung 9, ergänzt um den JavaScript Code für die

xPath evaluation, welcher Abbildung 40 aus Kapitel 17 X3D und xPath, entspricht, nur, dass der Kommentar „//DO SOMETHING“ durch das Toggeln des „render“ Attributes ersetzt wurde, was zum Ein- und Ausblenden des Objektes führt.

Der **html content** erzeugt die html Seite wie sie in Abbildung 41 zu sehen ist und enthält die ***X3D scene*** wie in Abbildung 9 beschrieben. Das aus Kapitel 11 „Gruppen und Referenzen“ stammende ***Group with DEF*** findet anschließend Anwendung. Der Bereich nach ***Complex Shape with texture*** ist analog zu Kapitel 10 „Komplexe Geometrie und Texturen“ und erzeugt das Haus aus Abbildung 31, das später referenziert und für jede Stadt eingeblendet wird.

Der Bereich simple ***shape with texture*** erzeugt eine Kugel mit derselben Textur wie in Abbildung 22 aus Kapitel 8 „Einfache Formen und Texturen“. Diese Kugel stellt die Erde dar, auf der die Städte eingezeichnet werden.

Im Folgenden werden nun Template Matches durchgeführt um Länder zu finden, die dann eine <group> Umgebung bilden. Dies ist in ***<xsl:template match="country">*** zu sehen. Danach werden die Städte gesucht, sowohl in Ländern, als auch in deren Provinzen.

Für jede gefundene Stadt wird in ***<xsl:template match="city">*** die „longitude“ ausgelesen und in das Bogenmaß umgerechnet und nur weiter verfahren wenn eine „longitude“ existierte.

Dann werden „longitude“ und „lattitude“ der Stadt in $***longitudeValue*** und $***latitudeValue*** gespeichert um dann zwei Rotationen und eine Translation hintereinander auszuführen wie in Kapitel 7.1 „Reihenfolge von Transformationen“ beschrieben. Dadurch werden die folgenden Objekte auf die „Erdoberfläche“ an ihre zugehörige geographische Position gebracht.

Als nächstes wird ein durchsichtiger Würfel erzeugt, der wie in Kapitel 15 „JavaScript und x3dom“ ein „onclick“ Attribut gesetzt bekommt, dass per JavaScript eine Nachricht erzeugt, die den Namen der Stadt ausgibt.

Als letztes wird das oben als Komplexe Form definierte Haus mittels des „USE“ Attribut referenziert wie in Kapitel 11 „Gruppen und Referenzen“ beschrieben.

19. Link Sammlung

http://www.blender.org/
Die Blender Homepage mit Downloadmöglichkeit und Tutorials

http://www.web3d.org/x3d/content/X3dTooltips.html
Die kommentierte DTD zu X3D

http://www.web3d.org/realtime-3d/x3d/what-x3d
Homepage des Web3D Konsortiums mit Definitionen zu X3D und vielem Zubehör

http://x3dgraphics.com/
Eine Seite, die einen guten Einstieg in X3D bietet und viele Beispiele mitbringt

http://www.x3dom.org/
Die x3dom Homepage mit vielen Beispielen für die Einbindung von X3D direkt im Browser und der aktuellen x3dom.js Version

Liste der Beispieldateien

00_show.xhtml

01_Basic_Example.xhtml

02_Simple_Shapes.x3d

03_Transform.x3d

04_Transformation_Order_1.x3d

05_Transformation_Order_2.x3d

06_Simple_Shapes_and_Textures.x3d

07_Complex_Geometry.x3d

09_Complex_Shapes_andTextures_1.x3d

09_Complex_Shapes_andTextures_2.x3d

10_Group_DEF_and_USE.x3d

11_Inline.x3d

12_javaScript.x3d

13_light.x3d

14_Viewpoint.x3d

Mondial XML to X3D.xhtml

to3d.xsl

mondial.xml

blender.x3d

Abbildungsverzeichnis