Zusatzübung 2 (mit X3D[1])

a. In Zusatzübung *1.b)* war ein 2D-Polygon in der *xy*-Ebene mit den 7 Vertices $P_0(1, 4.5, 0)$, $P_1(2.5, 1, 0)$, $P_2(4.5, 3, 0)$, $P_3(5, 1.5, 0)$, $P_4(6, 3.5, 0)$, $P_5(4, 5, 0)$, $P_6(2.5, 3.5, 0)$ in Dreiecke zu zerlegen. Stellen Sie dieses Polygon nun zuerst als X3D-IndexedFaceSet[2] dar, indem Sie auf Grundlage des nachfolgenden Code-Schnipsels das Feld „point" mit den Eckpunkten befüllen und das Feld „coordIndex" mit den zugehörigen Indizes. Beachten Sie, dass bei einem IndexedFaceSet jedes einzelne Polygon, d.h. *n*-Eck, mit „-1" abgeschlossen werden muss.

```
<Shape>
    <Appearance>
        <Material diffuseColor="0.7 0.5 0.3"></Material>
    </Appearance>
    <IndexedFaceSet convex="false" coordIndex="">
        <Coordinate point=""></Coordinate>
    </IndexedFaceSet>
</Shape>
```

Stellen Sie danach das in Dreiecke zerlegte Polygon als Knoten vom Typ IndexedTriangleSet[3] dar. Geben Sie dazu nacheinander die Indizes der Dreiecke in der richtigen Reihenfolge an. Im Gegensatz zum generischen IndexedFaceSet gibt es bei diesem Geometrieknoten nur ein einziges Feld namens „index" für alle Indextypen (entspricht sog. Single-Index-Rendering). Damit ist ein IndexedTriangleSet deutlich effizienter.
Hinweis: Szenengraph-Knoten können in X3D über DEF/USE geshared werden,[4] was evtl. zur Vermeidung von Copy and Paste für den Coordinate-Knoten interessant sein könnte.

b. Erzeugen Sie eine Box als IndexedFaceSet. Nutzen Sie Vierecke als Flächenprimitive. Das Zentrum der Box soll im Ursprung liegen und Länge, Breite, Höhe sollen je 2 Einheiten betragen.
Bei einem IndexedFaceSet kann man sowohl Flächen- als auch Vertexnormalen angeben. Geben Sie ebenfalls die je zugehörigen Normalenvektoren an, die analog zu den Vertexkoordinaten mit Hilfe eines weiteren Kindknotens vom Typ `Normal` (im Feld „vector") angegeben werden. Analog zu „coordIndex" besitzt das IndexedFaceSet ein Feld „normalIndex" zur Indizierung der Normalen, wobei Sie für diese Aufgabe aber keine Flächen- sondern Vertexnormalen nutzen sollen (Angabe pro Eckpunkt).

c. Fügen Sie Ihrer selbstgebauten Box ein rotes X3D-Material mit Glanzlicht hinzu. Ergänzen Sie die Szene noch um eine blaue Kugel, auch mit Glanzlicht. Betrachten Sie beide 3D-Objekte von allen Seiten. Warum sehen Sie ohne explizite Angabe einer Lichtquelle[5] überhaupt etwas?[6]
Schauen Sie sich das Tutorial https://doc.x3dom.org/tutorials/lighting/lights/index.html an.

Hinweis: Da X3D auch ein Szenengraph-System ist, läuft die Verschachtelung von Transformationen ähnlich wie in Qt. In X3D wird jedoch nicht zwischen Entities und Komponenten unterschieden, da hier alles ein „Node" ist, weswegen Transform-Knoten[7] u.a. weitere Transforms als Kinder haben können.

---

[1] Genau genommen wird hier das X3DOM-Framework (für X3D in HTML) genutzt: https://doc.x3dom.org/index.html
[2] https://www.web3d.org/documents/specifications/19775-1/V3.3/Part01/components/geometry3D.html#IndexedFaceSet
[3] https://www.web3d.org/documents/specifications/19775-1/V3.3/Part01/components/rendering.html#IndexedTriangleSet
[4] https://www.inf-schule.de/information/informationsdarstellungxml/darstellunginformation/fallstudie_3dgrafiken/exkurs/def
[5] https://www.web3d.org/documents/specifications/19775-1/V3.3/Part01/components/lighting.html
[6] https://www.web3d.org/documents/specifications/19775-1/V3.3/Part01/components/navigation.html
[7] https://www.web3d.org/documents/specifications/19775-1/V3.2/Part01/components/group.html#Transform

X3D

(Siehe https://www.x3dom.org/)

Geometrische Grundobjekte

Kugel: <sphere radius="1.0">

Zylinder: <cylinder radius="1.0" height="2.0">

Würfel: <box size="2.0 2.0 2.0">

Kegel: <cone bottomRadius="1.0" height="2.0">

Ring: <torus innerRadius="0.5" outerRadius="1.0">

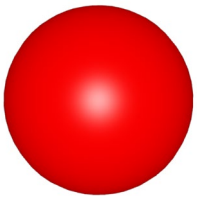
Ebenenstück: <plane size="2.0 2.0" subdivision="1 1">

Eigene Geometrien mit IndexedFaceSet

- Die Eckpunkte eines Polygons (hier "Face"), geg. als <Coordinate>
- Der Index auf einen solchen Eckpunkt (Vertex), geg. als Array: "coordIndex"
- Ende eines Polygons und Beginn eines neuen wird durch "-1" in Indexliste markiert
- Auf diese Weise lassen sich beliebig komplexe 3D-Objekte erzeugen
- Bsp. Rechteck-Geometrie:

```
<IndexedFaceSet coordIndex="0 1 2 3 -1">
    <Coordinate point="2 2 0, 7 2 0, 7 5 0, 2 5 0"></Coordinate>
</IndexedFaceSet>
```

Materialien[8]



- Material mit Transparenz: Feldwerte von *transparency* $\in [0,1]$
- Bsp.: Rotes Material mit hellem Glanzlicht auf Kugel

```
<Shape>
  <Appearance>
    <Material ambientIntensity='0.2' diffuseColor='1 0 0'
              specularColor='.4 .4 .4' shininess='0.8'>
    </Material>
  </Appearance>
  <Sphere radius='1.0'></Sphere>
</Shape>
```

Szenengraph (Gruppierung und Transformationen)

- 3D-Elemente werden hierarchisch angeordnet

[8] https://www.web3d.org/documents/specifications/19775-1/V3.2/Part01/components/shape.html#Material

- Ausgehend vom Wurzelknoten (d.h. vom <Scene>-Element) werden alle 3D-Elemente (z.B. <Shape>, <Box> usw.), je nach Knotentyp, als Kind- bzw. Geschwisterelemente in den Szenegraphen eingefügt
- <Transform>-Elemente helfen dabei, andere Objekte zu gruppieren sowie zu positionieren; solche Transformationsknoten können als Kindknoten beliebig viele Shape-Knoten oder auch andere Transformationsknoten haben

```
<transform translation="0 0 0" rotation="0 0 1 0" scale="1 1 1">
    ...
</transform>
```

- Transformationen in X3D
  - Verschiebung / Translation um $(1, 2, 3)^T$: <transform translation="1 2 3">
    - Default ist „0, 0, 0“
  - Skalierung um Faktor 1 in *x*, 2 in *y* und 3 in *z:* <transform scale="1 2 3">
    - Default ist „1, 1, 1“
    - Niemals und entlang keiner Achse mit 0 skalieren!
  - Rotation um Achse $(1, 0, 0)^T$ um 180°: <transform rotation="1 0 0 3.14159">
    - Axis-Angle Repräsentation der Rotation durch Angabe eines Einheitsvektors $\vec{r}$ als Drehachse sowie eines Drehwinkels α (letzter Wert)
    - Default ist „0, 0, 1, 0“
  - Reihenfolge, falls alle o.g. Attribute gesetzt sind:
    - Erst Skalierung, dann Rotation, zuletzt Translation: $p' = T \cdot R \cdot S \cdot p$
  - Weiteres Attribut: center
    - Gibt Rotationszentrum an (Translation entlang Rotationszentrum *c*, sog. Pivot Point); z.B. nützlich, wenn Objektzentrum nicht im Ursprung liegt
    - Falls Feld ‚center‘ auch gesetzt: $p' = T \cdot C \cdot R \cdot S \cdot C^{-1} \cdot p$

Einfaches Beispiel:

```
<X3D>
    <scene>
        <transform translation="-2 0 0">
            <shape>
              <appearance>
                <material diffuseColor="1 0 0"></material>
              </appearance>
              <box></box>
            </shape>
          </transform>
          <transform translation="2 0 0">
            <shape>
              <appearance>
                <material diffuseColor="0 0 1"></material>
              </appearance>
              <sphere></sphere>
            </shape>
          </transform>
    </scene>
</X3D>
```