X3D

Kazó Csaba 2007.

2007. november 30.

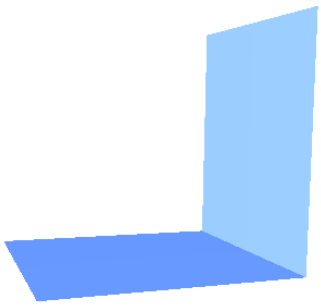
Az X3D egy virtuálisszíntér-leíró formátum. Valós idejű megjelenítést tesz lehetővé, így különösen alkalmas modellrekonstrukció eredményének vizualizációjára: a kapott modellt körbejárhatjuk, könnyedén megvizsgálhatjuk egy X3D megjelenítő segítségével.

Maga a formátum XML alapú, felépítését a következő egyszerű példafájl szemlélteti.

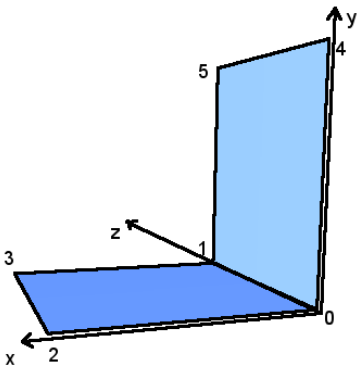
```
<?xml version="1.0"?>
<X3D version="3.0"
   profile="Immersive"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
   xsd:noNamespaceSchemaLocation=
      "http://www.web3d.org/specifications/x3d-3.0.xsd">
  <Scene>
    <Viewpoint position="1.3 .6 -2" orientation="0 1 0.06 2.75"
        fieldOfView=".6"/>
    <Background skyColor="1 1 1"/>
    <Shape>
      <Appearance>
        <Material emissiveColor=".3 .5 1"/>
      </Appearance>
      <IndexedFaceSet solid="false" coordIndex="0 1 3 2 -1 0 4 5 1 -1">
        <Coordinate point="0 0 0, 0 0 1, 1 0 0, 1 0 1, 0 1 0, 0 1 1"/>
      </IndexedFaceSet>
    </Shape>
  </Scene>
</X3D>
```

A gyökérelem tehát az `X3D`, magát a színtérleírást a `Scene` elemek fogják közre. A `Viewpoint` egy perspektív nézőpontot definiál, a `Background` a háttér színét állítja fehérre (az alapértelmezett háttér fekete). Objektumokat `Shape` elemmel definiálunk. Megadásuk két fő részből áll: a geometria és az anyagtulajdonságok leírásából. Utóbbi az `Appearance` blokkban szerepel, előbbire pedig számos lehetőségünk van, amint azt látni fogjuk.

A fenti kód az 1. ábrát állítja elő. Ezt az objektumot fogjuk példaként használni a továbbiakban.



1. ábra. A példa



2. ábra. X3D koordinátarendszer

0.1. Geometria megadása

Az X3D természetesen támogatja primitívek használatát (pl. Sphere, Cylinder, Box), de számunkra a pont alapú definíciók az érdekesek, hiszen a modellrekonstrukció ilyen formában szolgáltat eredményeket.

A 2. ábra a példaobjektumunk pontjait mutatja; ezeket megszámozzuk 0-tól 5-ig. Az ábrából az is látható, hogy az X3D jobbkezes koordinátarendszert használ. Ha az objektumunk éleit egységnyi hosszúnak választjuk, a 2. ábráról leolvashatjuk az egyes pontok pozícióját; pl. az 1. pont helye <0,0,1>, az 5. ponté <0,1,1> stb.

0.1.1. Pontábrázolás

Magukat a pontokat `PointSet` geometria megadásával jeleníthatjük meg:

```
<PointSet>
  <Coordinate point="0 0 0, 0 0 1, 1 0 0, 1 0 1, 0 1 0, 0 1 1"/>
</PointSet>
```

3. ábra. Pontfelhő megjelenítése

A `Coordinate` elem point attribútumában adjuk meg a pontok koordinátáit. Ezek hármasával csoportosítva értendők: az első három szám definiálja az első pontot, a második három a másodikat stb. A koordináták között szóköz és vessző egyaránt használható; a könnyebb áttekinthetőség érdekében érdemes a vesszőt tagolásra használni.

Az eredményt a 3. ábra mutatja. A `PointSet` pontjait az X3D nézőponttól függetlenül mindig egyetlen pixelen ábrázolja; az ábrán a pontokat utófeldolgozással tettem könnyen láthatóvá.

0.1.2. Vonalas megjelenítés

A LineSet elem felhasználásával töröttvonalakból építhetjük fel az objektum geometriáját:

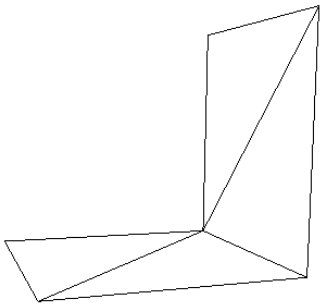
```
<LineSet vertexCount="8 3">
  <Coordinate point="0 0 0, 0 0 1, 1 0 1, 1 0 0, 0 0 0,
                0 1 0, 0 1 1, 0 0 1, 0 1 0, 0 0 1, 1 0 0"/>
</LineSet>
```

A `Coordinate` elem point attribútuma a vonalakat alkotó pontokat tartalmazza, a `vertexCount` attribútum pedig azt adja meg, hogy az egyes töröttvonalak hány pontból állnak. Példánkban a `Coordinate` elemben 11 pontot adtunk meg, ezekből két töröttvonalat állítunk elő: az első nyolc pont a körvonalat, a maradék három pont a két átlót alkotó töröttvonalat definiálja (4. ábra).

Ugyanez a geometria másképpen is megadható az `IndexedLineSet` elem segítségével. Ekkor minden pontot elegendő egyszer leírni a `Coordinate` elemben, a töröttvonalakat pedig csupán a megfelelő pontok sorszámának felírásával adjuk meg. A diszjunkt töröttvonalakatat -1-es számmal választjuk el egymástól.

Az alábbi kódrészletben a pontokat a 2. ábrán feltüntetett sorrendben definiáljuk, így az ábráról leolvasható indexekkel hivatkozhatunk rájuk. Szemantikailag a fenti `LineSet` definícióval teljesen ekvivalens leírás:

```
<IndexedLineSet coordIndex="0 1 3 2 0 4 5 1 -1, 4 1 2">
  <Coordinate point="0 0 0, 0 0 1, 1 0 0, 1 0 1, 0 1 0, 0 1 1"/>
</IndexedLineSet>
```



4. ábra. Vonalháló megjelenítése

0.1.3. Háromszögháló megadása

Háromszögekből álló geometria definiálására számos elem használható. A következő példák mindegyike az 1. ábrát állítja elő.

A legegyszerűbb konstrukció `TriangleSet`: itt a `Coordinate` elemmel megadott pontok hármasával fognak egy-egy háromszöget alkotni. Tehát az első három pont (az első kilenc koordináta) az első háromszög, a következő három pont a következő háromszög stb. A `solid` attribútum beállításával azt biztosítjuk, hogy a háromszögek mindkét oldalról láthatóak; alapértelmezés szerint a háromszög hátoldalát nem látjuk (backface culling).

```
<TriangleSet solid="false">
  <Coordinate point="
     0 0 0, 0 0 1, 1 0 0,
     0 0 1, 1 0 0, 1 0 1,
     0 0 0, 0 1 0, 0 0 1,
     0 1 0, 0 0 1, 0 1 1"/>
</TriangleSet>
```

A geometriát itt négy háromszög alkotja; a 2. ábra sorszámozásával ezek a (0,1,2), (1,2,3), (0,4,1) és (4,1,5) ponthármasok által meghatározott háromszögek. A vonalábrázolásnál megismert indexelés segítségével a háromszöghalmaz megadását is rövidíthetjük:

```
<IndexedTriangleSet solid="false" normalPerVertex="false"
    index="0 1 2, 1 2 3, 0 4 1, 4 1 5">
  <Coordinate point="0 0 0, 0 0 1, 1 0 0, 1 0 1, 0 1 0, 0 1 1"/>
</IndexedTriangleSet>
```

A háromszögháló színezését meghatározó normálvektorok generálása jelen esetben automatikusan történik. A `normalPerVertex` opció kikapcsolásával azt érjük el, hogy egy háromszög mindhárom csúcspontjában azonos, az adott háromszögre merőleges legyen a normálvektor. Alapértelmezés szerint ugyanis egy csúcsponthoz tartozó normálvektor a hozzá illeszkedő háromszögek normálvektorainak átlagaként állna elő, de nekünk jelen esetben nem ez a szándékunk.

(Ráadásul például az első két háromszöget különböző körüljárási iránnyal adtuk meg, ami azt jelenti, hogy a normálvektoruk ellentétes irányú. Ebben az esetben az átlagolás következtében a két háromszög által alkotott lap árnyalása nem lenne egyenletes.)

A fenti definíciók külön háromszögeket definiáltak. A következő két konstrukció lehetővé teszi bizonyos háromszögcsoportok tömörebb leírását. Ezeknek is van indexelt és nem indexelt változata; a rövidség kedvéért csak az előbbit fogjuk feltüntetni.

A `TriangleStripSet` ill. `IndexedTriangleStripSet` esetén a pontsorozatból a háromszögek a következőképpen adódnak: az első három (a 0., 1. és 2. sorszámú) pont alkotja az első háromszöget, az 1., 2. és 3. pont alkotja a másodikat, a 2., 3. és 4. a harmadikat és így tovább. (A pontsorozatot nem indexelt esetben maga a `Coordinate` elem írja le, indexelt esetben az indexek határozzák meg.) Tehát az egymást követő háromszögeknek egy-egy oldaluk közös kell hogy legyen. Ezek az X3D elemek lehetővé teszik azt, hogy több, különálló „háromszögcsíkot" definiáljunk egyszerre, de a mi példaobjektumunk egyetlen csíkból összerakható:

```
<IndexedTriangleStripSet solid="false" normalPerVertex="false"
    index="2 3 0 1 4 5">
  <Coordinate point="0 0 0, 0 0 1, 1 0 0, 1 0 1, 0 1 0, 0 1 1"/>
</IndexedTriangleStripSet>
```

Itt tehát négy háromszög jön létre, a (2,3,0), (3,0,1), (0,1,4) és (4,0,5) indexű ponthármasok között.

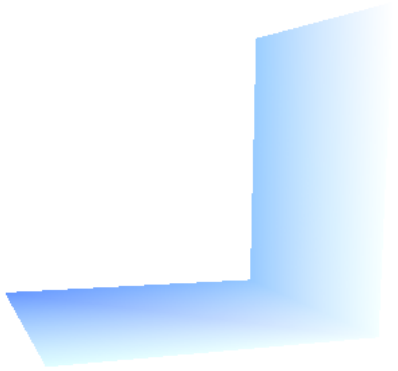
Végül a `TriangleFanSet` ill. `IndexedTriangleFanSet` segítségével is megadhatunk háromszöghálót. Ekkor a pontsorozat következő elemei fognak háromszöget alkotni: a 0., 1. és 2., a 0., 2. és 3., a 0., 3. és 4. stb. Tehát minden egyes újabb háromszöghöz csak egyetlen új csúcsot adunk meg; a másik két csúcs az elsőként illetve a legutoljára megadott pont lesz. Itt is több ilyen „legyezőt" definiálhatunk egyszerre, de most is elegendő egyetlen sorozat, ha a legyező középpontjának a 0. (origóbeli) pontot választjuk:

```
<IndexedTriangleFanSet solid="false" normalPerVertex="false"
    index="0 2 3 1 5 4">
  <Coordinate point="0 0 0, 0 0 1, 1 0 0, 1 0 1, 0 1 0, 0 1 1"/>
</IndexedTriangleFanSet>
```

Ugyanezt az eredményt kapjuk, ha az `index` attribútumban megadott pontsorozatot a következőre változtatjuk: 2 3 1 0 -1 4 5 1 0. Ez két legyezővel állítja elő az objektumot: az első az alsó négyzetet hozza létre (középpontja a 2. pont), a második pedig az oldalsót (középpontja a 4. pont). Az indexsorozatban szokás szerint -1-et használunk a komponensek elválasztására.

0.2. Normálvektorok

A fenti példákban a geometria háromszöghálós definiálása esetén az X3D megjelenítő automatikusan generál normálvektorokat minden csúcshoz. A



5. ábra. Árnyalás megadott normálvektorokkal

`normalPerVertex` attribútummal szabályozhatjuk, hogy a generált normálvektorok a szomszédos háromszögek normálvektorainak átlagaként adódjanak-e (ez tulajdonképpen minden élet automatikus elsimítva árnyal), vagy háromszögenként egységesek legyenek-e (ekkor sehol nincsen simítás).

Az automatikus generálást egy kicsit finomabban szabályozhatjuk a `creaseAngle` attribútummal. Ha ennek megadunk egy radián értéket, akkor két szomszédos háromszög között akkor lesz sima árnyalás, ha a geometriai normálvektoruk által bezárt szög a megadottnál kisebb.

Természetesen mi magunk is megadhatjuk a normálvektorokat a `Coordinate` elemhez nagyon hasonló `Normal` elem vector attribútumában. Az egyes vektorok a `Coordinate` elem megfelelő pontjaihoz tartoznak. Az alábbi definíció például egyrészt kisimítja a 0. és 1. csúcs közötti élt, másrészt a hozzánk közelebbi, 2., 0. és 4. csúcsokhoz tartozó normálvektorokat a nézőpont felé fordítja, így egy meghajlított hatást érve el (5. ábra). Persze ennél a nagyon egyszerű, kevés háromszögből álló objektumra a hatás nem igazán látványos.

```
<IndexedTriangleFanSet ccw="false" solid="false"
    index="0 2 3 1 5 4">
  <Coordinate point="0 0 0, 0 0 1, 1 0 0, 1 0 1, 0 1 0, 0 1 1"/>
  <Normal vector="1 1 -1, 1 1 0, 0 1 -1, 0 1 0, 1 0 -1, 1 0 0"/>
</IndexedTriangleFanSet>
```

A `ccw` attribútummal meg kell adnunk, hogy jelen esetben a normálvektorokkal ellentétes irányból nézve a háromszögek körüljárási iránya az óramutató járásával megegyező (az alapértelmezés a `ccw='true'`, vagyis az óramutató járásával ellentétes irány).

0.3. Textúrázás

Természetesen lehetőség van arra is, hogy egy képről textúrázzuk az objektumot. Ehhez egyrészt meg kell adnunk a képfájl elérési útját az `Appearance` blokk

6. ábra. Textúrázott háromszögháló

`ImageTexture` elemében (tetszőleges URL-eket megadhatunk, tehát a fájl származhat az Internetről). Másrészt a geometria definícióját ki kell egészítenünk a textúrakoordináták megadásával. Ezeket a `TextureCoordinate` elemben soroljuk fel; egy az egyben megfelelnek a `Coordinate` elemben felsorolt térbeli pontoknak. A textúrasíkon a képet úgy képzeljük el, hogy bal alsó sarka van az origóban, jobb felső sarka pedig az (1,1) koordinátájú pontban; ennek megfelelően definiáljuk a textúrakoordinátákat:

```
<Appearance>
  <ImageTexture url="../img/moscow.jpg"/>
</Appearance>
<IndexedTriangleFanSet solid="false" normalPerVertex="false"
    index="0 2 3 1 5 4">
  <Coordinate point="0 0 0, 0 0 1, 1 0 0, 1 0 1, 0 1 0, 0 1 1"/>
  <TextureCoordinate
    point=".8 .5, .2 .5, .8 .1, .2 .1, .8 .7, .2 .7"/>
</IndexedTriangleFanSet>
```