Modellezés OpenBVE-ben, azaz tereptárgyak létrehozása

Készítette: Krisz, 2006
Második, átdolgozott kiadás, 2014-2017
ckirbi@freemail.hu

A második kiadás elkészítésében közreműködött: Phonteus Nevolius.

Legutóbbi módosítás: 2017. október 6.

Bevezetés

Ez a leírás azért született meg, mert már túl sok helyről kérték, hogy csináljak ezt-azt, többen kérdezték, hogyan működik az objektumkészítés. Nos számukra jó hír: nem misztikum.

OpenBVE modelleket bárki gyárthat. Nem kell hozzá más, mint egy kis tehetség és a modell bonyolultságától függő mennyiségű idő. De ne riasszon el senkit sem, ha ez a leírás bonyolultnak tűnik vagy hosszúnak. Nem lehet pár perc alatt megtanulni modellezni. Az itt leírt példákat nagyon ajánlott begépelni (vagy Ctrl-C – Ctrl-V technikával bemásolni a Jegyzettömbbe), majd megnézni Object Viewerrel. Csak így lehet megérteni, melyik parancs mit csinál, hogyan működik.

Ha valaki mégis úgy gondolja, hogy a jegyzettömbös verzió túl durva neki, annak ajánlom figyelmébe a 3D-s tervező programokat, nagyon sok ingyenes és jól használható alkalmazás van, le kell tölteni ezeket a webről. Ez a leírás viszont nem tér ki arra, hogy ezekkel a programokkal hogyan kell modellezni. Ennek több oka is van: egyrészt minden tervező program más (az alapoktól eltekintve), másrészt ezek működése viszonylag bonyolultabb, hogy ebbe a leírásba beleférjen. Ehhez is segítséget lehet találni a neten. Ha mégis ezzel terveznél, akkor egy fontos dolog: a modellt DirectX (.X) formátumban kell elmenteni, mert a játék csak ezt a formátumot támogatja a B3D és a CSV mellett.

Mi kellhet még?

Ez attól függ, milyen modellt készítünk el. Ami mindenképpen jól jöhet: az Object Viewer, ez letölthető a hivatalos OpenBVE oldalról más fejlesztőeszközökkel együtt *Developer's Tools* néven. Létezik még egy objektummegjelenítő program, a Structure Viewer, ám ez a BVE-hez készült és az OpenBVE-ben megjelent új funkciókat nem támogatja, valamint bizonyos hibákat nem jelez, így a használata néhány különös esettől eltekintve nem ajánlott. A Structure Viewer 2.x-es verziója exportálni is tud DirectX formátumba, amit szerkesztő programmal meg tudunk nyitni. Aztán egy jellegrajz vagy tervrajz is jól jön, lehetőleg minél pontosabb legyen. Egy számológép is elkél majd a koordináták számolásához. A modell textúrázásához fényképek, lehetőleg szemből fényképezve (tehát ne szögből). A fényképek ne legyenek se túl sötétek, se nagyon világosak, se életlenek; mindenkinek belátására bízom, milyen minőséget talál elfogadhatónak. Annyit azért elöljáróban, hogy a távolban elhelyezendő objektumoknak nem szükséges részletgazdagnak lennie, azonban azokhoz a tárgyakhoz, amik a sínek mellett lesznek, nem árt jó minőségű textúrát szerezni. Hangsúlyozom: a modell értékét nem csak az határozza meg, hogy milyen textúrával van ellátva, hanem az is, hogy maga a modell hogy néz ki. Ezalatt azt értem, hogy mindenki képzeljen el egy olyan autót, aminek minden oldala négyzet alakú. Ugye furcsa, hogy szemből ránézve a modellre a motorháztető és a szélvédő egy vonalban van…

Geometria

Nézzünk bele egy kicsit a matematika könyvbe, a geometria címszónál! Ezeket a fogalmakat fontos megérteni, sőt megtanulni, de remélem, nem okoz majd problémát. Mindazoktól, akik matematikával foglalkoznak, elnézést kérek, ha a definíciók nem pontosak, sőt kicsit „buták”.

Tér

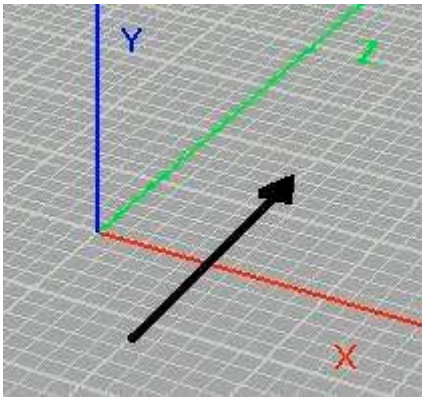
Mivel a játék a 3 dimenziós térben folyik (értsd te is a 3D-s térben élsz), ezért fontos megérteni. Ez tudom, hogy triviális, de ha körbenézel most a szobádban, mit látsz?! 3 dimenziós tárgyakat. Azonban a tárgyak nem láthatóak teljes egészükben. Ezalatt azt értem, hogy hiába látod a monitorod képernyőjét, ebből a szögből nem látod a hátulját, vagy például nem látsz a hátad mögé. Ez utóbbi azért van, mert a tárgyak nem esnek a látószögedbe (kevesebb mint 180 fok). Ráadásul vannak tárgyak, amiket (mozdulatlan fej- és szemállással) élesen látsz, vannak, amik homályosan, de azért megjelennek a látómeződben, és vannak, amit egyáltalán nem látsz. Ez ugyanígy van a játékban is, lesznek olyan pontok, ahol a tárgyaknak nem érdemes olyan részletesnek lenni.

Pont és vektor

A térben minden pont távolságát/helyét meg tudjuk adni a kiindulási pontból 3 szám segítségével. Ezt a rendezett számhármast röviden az aktuális pontból a végpontba húzott szakasz (vektor) koordinátáinak fogjuk hívni.

Koordináták, koordinátarendszer

A BVE modelleknél a normál Descartes-koordinátarendszert használjuk, amit az ábrán láthatunk.



Képzeljük el magunkat olyan pozícióban, hogy ezt a 3 egymásra merőleges színes vonalat látjuk, és a metszésponton állunk. Ekkor a zöld Z-vel jelzett egyenes iránya lesz a menetirány, tehát a sínek is ilyen irányban futnak. A piros X jelenti a bal-jobb irányítottságot. A kék Y a fel-le-t. Ahogy az előzőekben említettem, egy pontot ebben a rendszerben 3 szám határoz meg: (X,Y,Z). A (0,0,0) számhármast az origónak hívjuk. Tekintsük ezt most a talpunk közepének (ha két lábon állunk). A számok valós számok, lehetnek negatívak, pozitívak, és 0. Ha az X értéke negatív, az azt jelzi, hogy a kezdőponttól (origó=talp közepe) balra helyezkedik el a pont, ha pozitív, akkor

jobbra. Az Y értéke ha pozitív, akkor a föld felett van (pl. 1.7-1.8 esetén mondjuk szemmagasságban), ha negatív, akkor a talpunk alatt. Mielőtt mindenki azt hinné, hogy ennek nincs értelme, tiltakoznék. Igenis van! Ha eljutsz majd a Translate parancsig, máris meglátod az értelmét. A Z értékét már ki is lehet találni: ha pozitív, akkor a fejünk előtt van a pont, ha negatív, akkor az origó mögött, tehát a hátunk mögött. Egy tárgy elkészítésekor (jó esetben!) felmerül majd a kérdés: *origókezdetű* vagy *origóközéppontú* legyen-e a tárgy (saját elnevezések). Az előbbi esetben a tárgy legelülső pontja (tehát amit szemből látunk, pl. egy peron) legfeljebb az origó, de inkább mögötte van, tehát a Z koordináták nemnegatívak. Ez akkor hasznos, ha olyan tárgyat készítünk, amit feltehetőleg később nem forgatva helyeznek el a pályában, ilyen pl. egy peron a sín mellett. Ez azt teszi, ha én azt mondom, hogy a tárgy eleje, pl. 250 méternél legyen, akkor az ott is lesz. Ellentétben az origóközpontú esettel. Ilyenkor a tárgyat úgy készítjük el, hogy a középpontja az origó legyen. Ez középpontosan szimmetrikus tárgyak esetén jól jön, pl. fák vagy egy emeletes ház. Ilyen esetben a ház tengelye nem tolódik el az origóból forgatás esetén, viszont ha pl. egy 20 egység hosszú házat így készítünk el, akkor az eleje −10-nél lesz. Meglátás kérdése, hogy melyiket alkalmazzuk, de azért gondoljunk majd a pályakészítőkre is! Fontos: a vasúti járműveket, beleértve mozdonyt, személykocsit, teherkocsit, villamost, metrót és minden más sínen guruló dolgot mindig origóközéppontúan kell elkészíteni.

Akkor gyakorlásnak itt van pár koordináta, mindenki gondolja végig merre található a pont (origónak tekintsük megint a talpunk közepét) – megoldások a leírás végén. Hangsúlyozom, ezt elemi fontosságú megérteni!

1. feladat

a) (0.3, −0.5, 2)
b) (−2.1, 0, 4)
c) (1, 1, −3)
d) (0, 1.5, 0)

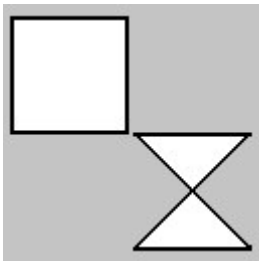
Nem írtam, de gondolom mindenki kitalálta, a számok méterben értendők, tehát pl. a *0.2* 20 centimétert, az *5* pedig 5 métert jelent. A tizedesvessző helyett pontot használunk.

Vonal

Egy szinttel feljebb lépünk. Vannak pontjaink, össze tudjuk kötni őket. Két pontot ha összekötünk, kapunk egy szakaszt (vonalat). Ez egyszerű.

Felület, sík

Jelöljük a pontokat rendre P_0, P_1, P_2,..., P_n −nel. Ha ezeket úgy kötjük össze, hogy P_0-P_1, P_1-P_2,..., P_n-P_0 akkor egy felületet kapunk. Természetesen nem mindegy, hogy milyen sorrendben kötjük össze a pontokat. Itt egy példa 4 pontra:



Tehát láthatjuk, hogy ugyanazokra a pontokra más-más összekötéssel más-más ábrát kaptunk. Természetesen 4 pontra nem csak ez a két alakzat jöhet ki, lehet próbálkozni.

Ezeket a síkbeli felületeket poligonnak hívjuk (tehát sokszögek). Arra nincs kikötés, hogy egy poligon hány pontból állhat, de azt gondolom mindenki belátja, hogy minimum 3-ból. Ráadásul az OpenBVE megjelenítésénél látni fogjuk, hogy bizonyos konvex-konkáv összekötéseknél nem a várt eredményt kapjuk. Csak emlékeztetőül: konvex az a síkidom, melynek bármely 2 pontját összekötő szakaszt tartalmazza a síkidom (lásd a fenti négyzet). Konkáv: ami nem konvex (lásd a „homokórát").

A síkot nem definiálnám, mindenki tudja, hogy például egy gúla (piramis) csúcsai nem esnek egy síkba, viszont pl. egy asztalon fekvő papírlap mind a 4 sarokpontja (természetesen az összes pontja is) egy síkban vannak. A lényeg: két egymással szöget bezáró felületet mindenképpen majd külön felületként kell létrehozni.

Ezzel végeztünk is, már csak annyit kell tudnunk, hogy felületekből tudjuk összerakni a testeket (objektumokat).

Végre modellt készítek...

Előkészületnek indítsuk el az Object Viewert (OV). Ez fogja megjeleníteni az elkészített objektumot. Néhány szót a kezeléséről, tényleg röviden: az F7-tel tudunk egy modellfájlt megnyitni. A számbillebntyűzet 2, 4, 8 és 6 gombjaival tudjuk mozgatni a képzeletbeli kamerát a modell körül, a nyilakkal pedig forgathatjuk a kamerát. Az egér bal gombját lenyomva tartva az egeret mozgatva is tudjuk mozgatni a kamerát, a jobb gombbal pedig szintén forgathatjuk a kamerát. Fontos: ez csak a megjelenítésre vonatkozik. F5-tel újrafrissíthetjük az ablakot, Del-lel kitörölhetjük a képet. Ennyit a kezelésről, használat közben könnyebb rájönni a mikéntre.

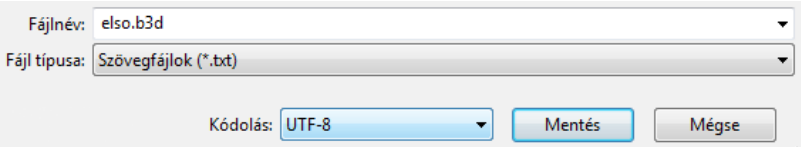
Kétféle szöveg modellfájl létezik OpenBVE-hez: a régi típusú B3D és az újabb CSV. Nagy különbség nincs köztük, sőt át is lehet konvertálni az egyiket a másikba. Én jobban szeretem a B3D típust, úgyhogy erre térnék ki, de az internetről letölthető programmal átalakíthatjuk az egyik formátumú fájlt a másikba és vissza.

Nyissuk meg a Jegyzettömböt! Minden parancsot külön sorba írjunk! Ha megjegyzéssel akarjuk tarkítani a fájlt, akkor azt a ';'-vel tehetjük meg. A pontosvessző után található szöveget a program nem veszi figyelembe. Itt jegyzem meg, a program nem érzékeny a kis-nagybetűkre, azaz a *vertex* és a *VERTEX* ugyanazt jelentik.

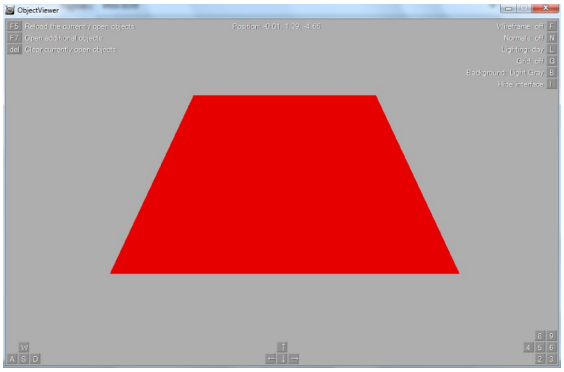
Akkor első körben hozzunk létre egy sima négyszöget (legyen trapéz) velünk szemben. Ehhez gépeljük be a következőket:

```
;Első BVE objektumom
;Készítette: én
[MeshBuilder]
Vertex -1,2,0
Vertex  1,2,0
Vertex  2,0,0
Vertex -2,0,0
Face 0,1,2,3
Color 255,0,0
```

Mentsük el elso.b3d néven. Fontos, hogy UTF-8 karakterkódolással mentsük, ezt a *Mentés másként* ablakban tudjuk kiválasztani.



Mentéskor és a fent említett módon nyissuk meg az OV programmal. Ekkor ráközelítve ezt kell hogy lássuk:



Rendben is van, trapézról volt szó, az lett. Az egér segítségével járjuk körbe (a számbillentyűzet 6-os gombja és a balra nyíl együttes lenyomásával könnyen tudjuk mozgatni a kamerát a trapéz körül) – egyszer csak azt tapasztaljuk, hogy az objektum eltűnik. Nem kell megijedni, ez egy egyoldalas poligon, de természetesen megoldható, hogy hátulról is látható legyen.

Akkor nézzük meg, mit is írtunk be. A fájl első két sorát nem magyarázom, ez **comment**, ahogy említettem fent. Ide be lehet írni a készítő nevét, az objektum nevét, méreteit stb. illetve később ilyen commentekkel segíthetünk magunknak, hogy egy adott felület mit is „csinál” az egész objektumban (pl. „ütköző” vagy „tartály”, stb.).

A **[MeshBuilder]** parancs egy új szekciót jelent. Ez mondja meg a programnak, hogy most új felületet hozzon létre a megadott pontokból. Azt is lehet, hogy több felületet hozunk létre egy MeshBuilder szekcióban. Nemsokára erre is látunk példát. Továbbá külön felületnek kell létrehozni olyan egy síkban levő felületet, ahol a konvex-konkáv részek miatt ezek a programban rosszul lennének összekötve, így nem a várt végeredményt kapjuk. Ilyen pl. egy ötágú csillag esetében, hiába esnek a csillag sarkai egy síkba és mondjuk meg a programnak milyen sorrendbe

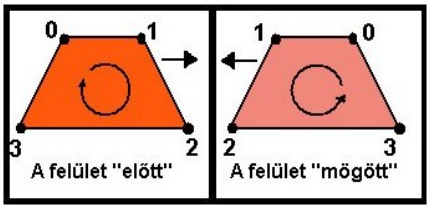
kösse össze a pontokat, zagyvaságot kapunk (ki lehet próbálni).

Ezután következnek a **Vertex X,Y,Z** parancsok. Ezek a pontok koordinátáit jelentik az origóhoz képest (vertex=pont). A paraméterek a fent ismertetett koordináták. A pontok azonosítása egyszerű számozással történik. Tehát a MeshBuilder sort követő első Vertex x,y,z sor azonosítja a **0.** (!) pontot, a következő az elsőt, stb. Így kiszámolható, hogy az utolsó pont száma a 3-as. Általánosságban n pont esetén a pontok számozása 0-tól (n-1)-ig terjed. Rendben, akkor mit is jelentenek a számok? Aki megcsinálta és megértette az 1-es példát fentebb, annak biztos nem fog gondot okozni ez: van a 0-s pontunk ami balra van −1 méterre, 2 méter magasan, és 0 méter mélyen. Az 1-es pont jobbra 1 méterre, 2 méter magasan, 0 méter mélyen. A 2-es pont 2 méterre jobbra, 0 méter magasan (a talajon) és 0 méter mélyen. Végül a 3-as pont 2 méterre balra, 0 méter magasan (a talajon) és 0 méter mélyen. Le is rajzolhatjuk magunknak egy papírra, és tényleg azt kapjuk, hogy a 0 a bal felső sarok, az 1 a jobb felső sarok, a 2 a jobb alsó sarok, a 3 a bal alsó sarok koordinátáit jelenti.

Hogyan kötnénk össze a lerajzolt pontokat, hogy a fenti képet lássuk? Egyszerű: a 0-1-2-3 egy lehetséges megoldás. Természetesen jók az 1-2-3-0, 2-3-0-1, 3-0-1-2 (tehát az óramutató járásával egyirányú) összekötések is. Gondolhatnánk: de hát úgy is összeköthetjük, hogy 0-3-2-1 stb, tehát az óramutató járásával ellentétes irányban. Papíron ez rendben is van!

Lássuk akkor a B3D megjelenítését! Ki lehetett találni, az összekötés sorrendjét a Face 0,1,2,3 parancs mondja meg. Ez így is van. Ezzel adjuk meg pontosan EGY, SÍKBELI felület pontjainak összekötési sorrendjét. Változtassuk meg most akkor a 0,1,2,3-at pl. 1,2,3,0-ra, és nézzük meg az eredményt! Nincs változás, nem meglepő.

Akkor most változtassuk meg a 3,2,1,0-ra! Azt tapasztaljuk, hogy látszólag eltűnt a felületünk. Ez azonban nem igaz. Járjuk körbe az objektumot, és azt tapasztaljuk, hogy a hátoldalon ott van a felület. De miért? Jogos a kérdés: azt kell tudni, hogy egy adott nézőpontból azok a felületek láthatóak a BVE-ben, amelyek pontjainak az összekötése az óramutató járásával azonos irányban történt. Fontos, hogy ez függ az adott nézőponttól. Ezért van az, hogy míg a normál esetben, ha az origó mögül nézzük a trapézt, akkor a 0,1,2,3 összekötés megfelel az óramutató járásának irányával. Ha viszont forgunk, és a trapéz mögé kerülünk, ebben az esetben ez viszont ez ellentétes az óramutató járásával, amint a képen is látszik. A fekete nyilak a pozitív X irányt jelentik.

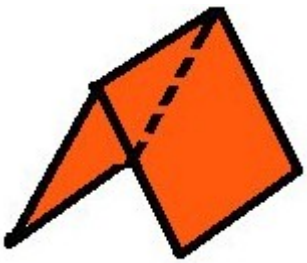


Most már remélem érthető miért „tűnik el” az objektum forgás közben. Kérdés: nem lehet megoldani, hogy egy felület mindkét irányból látható legyen? Dehogynem, ekkor a Face parancs helyett a **Face2** parancsot kell használni, melynek paraméterei ugyanazok, csak ez kétoldalú poligont hoz létre. Tessék kipróbálni!

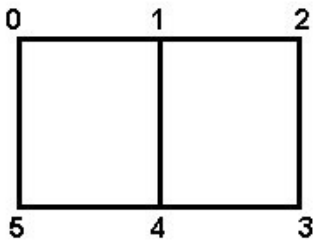
Végezetül már csak egy parancs van amivel nem foglalkoztunk: ez a **Color R,G,B**. Ezzel az utasítással lehet kiszínezni egy adott felületet. Fontos, hogy egy adott MeshBuilder szekcióban létrehozott összes felület ugyanezt a színt kapja meg. Az R,G és B paraméterek rendre a piros, a zöld és a kék színösszetevőket jelentik, azaz ezekből az alapszínekből keveri ki a program az adott színt. A 0 értéknél az adott színből nem használ fel, 255 esetén pedig teljesen belekeveri a végső színbe. A kettő között fokozatosan van elosztva a mennyiség, így különböző értékekre különböző árnyalatokat tudsz kikeverni. A (0,0,0) pl. a fekete színt, a (255,255,255) a fehéret, a (255,0,255) a lilát jelenti stb. Egy neked tetsző szín összetevőit könnyen megtudhatod például a Paint segítségével, méghozzá Windows XP-n a Színek/Színek definiálása menüben az Egyéni színek definiálása gombbal érhető el, Windows 7-en és 8-on pedig egyszerűen a Színek szerkesztése gombra kattintva. Itt a jobb oldali panelen kiválasztod a kívánt árnyalatot és a program kiírja a piros/zöld/kék összetevők értékeit. Ezeket feljegyzed, és ilyen sorrendben írod majd be a Color utasítás után.

A Color utasításnak lehet egy negyedik paramétere, ezt Alpha-nak hívják. Ezzel a paraméterrel a felület „áttetszőségét” lehet állítani, értéke ugyanúgy 0 és 255 közé esik. Minél kisebb az értéke, annál átlátszóbb lesz a felület, 0-nál nem is látszik. 255 esetén pedig egy matt, nem átlátszó felület jön létre. Ezzel kiválóan lehet pl. ablaküvegeket létrehozni. Ha elhagyjuk a Color végéről, az automatikusan 255 értéket jelent. Na akkor most lehet és kell is játszani az értékekkel! Változtassuk meg a Vertex-nél a koordinátákat, de csak ésszel (pl. egy 500 méter x 500 méter kiterjedésű felület túl nagy, egy 0.00005×0.00005 pedig túl kicsi) Adjunk a felületnek más színt (legyen átlátszó, stb.)!

Lépjünk egy kicsit tovább! Amint említettem, egy MeshBuilder szekcióba több felületet is létre lehet hozni, amelyek nincsenek egy síkban, egymással szöget zárnak be, de vannak közös pontjaik. Klasszikus példa erre a sátortető, vagy a kinyitott könyv felülről, amint ezen a sematikus ábrán is látható:



Ez síkban kiterítve így néz ki a csúcsok beszámozásával, a csúcsok számozása önkényes.



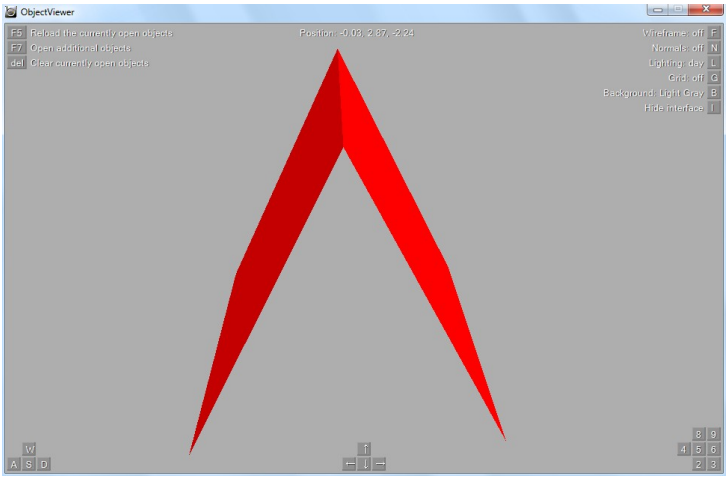
Remélem most már mindenki tudja, milyen sorrendben kell összekötni a csúcsokat: a bal oldali téglalapot 0,1,4,5 (az 1,4,5,0 illetve 4,5,0,1 és 5,0,4,1 is jó megoldások), míg a jobb oldalit 1,2,3,4

sorrendben (vagy 2,3,4,1 / 3,4,1,2 / 4,1,2,3). Ha valakinek ez így nem tiszta annak javaslom vegyen elő egy papírlapot hajtsa félbe és számozza be a sarkokat, illetve a félbehajtásnál a fenti sorrendben. Forgassa a papírt mint a fenti pirossal színezett ábra és olvassa le a számokat az óramutató járásával egyezően. A fenti számsorrendet kell kapni.

Így már könny megírni a B3D fájlt is:

```
;Sátortető
;Készítette: én
[MeshBuilder]
Vertex -1,0,2
Vertex 0,2,2
Vertex 1,0,2
Vertex 1,0,0
Vertex 0,2,0
Vertex -1,0,0
Face 0,1,4,5
Face 1,2,3,4
Color 255,0,0
```

A végeredmény pedig:



Ha ezek megvannak, akkor itt van egy újabb feladatsor, amit ugyancsak ajánlott megcsinálni. Ha valakinek nem megy, a leírás végén ott vannak a megoldások, az alapján gondolja végig mi az ami nem sikerült, és próbálkozzon újra!

2. feladat

a) Hozz létre egy olyan négyzetet, amelynek oldalai a Z és az Y egyenessel azonosak (azaz a YZ síkba esik). Legyen a négyzet oldala 2,5 méter hosszú. Elhelyezkedése: a középpontja legyen 0 méter magasan és 0 méter mélyen! Mindkét oldalról legyen látható, a színe legyen narancs és legyen egy kicsit áttetsző!

b) Készíts egy templomtornyot vagy piramist (4 db egybevágó háromszögből áll), a háromszögek alapja legyen 1.5 méter. A magassága legyen 3 méter, a piramis aljának (legyen neki!) a közepe pont az origóban álljon. Mindkét oldalról legyenek láthatóak az oldalai. Fesd ki kékre, és legyen áttetsző (mondjuk 80-as értékkel)!

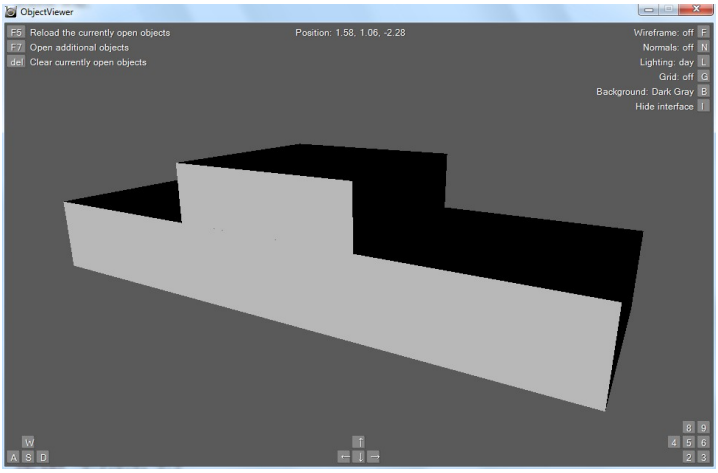
Komplexebb objektumok

Ha az előző fejezeten sikerrel túljutottunk, nézzünk egy „összetettebb” tereptárgyat! Olyat, amelyben nem tudjuk a felületeket egy MeshBuilder szekcióban létrehozni – például azért mert egyáltalán nincsenek közös pontjaik vagy a különböző felületeket más-más színnel szeretnénk ellátni stb. Mi ilyenkor a teendő?! Egyszerű, ezeket külön MeshBuilder szekcióba kell írni, egymás alá. Egy példa erre: egy dobogó, melynek elülső és hátsó oldala szürke, a többi oldala fekete lesz.

```
;Dobogó objektum
;Készítette: én

[MeshBuilder]
Vertex -1.5,0,-0.5
Vertex -1.5,0.4,-0.5
Vertex -0.5,0.4,-0.5
Vertex -0.5,0.7,-0.5
Vertex 0.5,0.7,-0.5
Vertex 0.5,0.4,-0.5
Vertex 1.5,0.4,-0.5
Vertex 1.5,0,-0.5

Vertex 1.5,0,0.5
Vertex 1.5,0.4,0.5
Vertex 0.5,0.4,0.5
Vertex 0.5,0.7,0.5
Vertex -0.5,0.7,0.5
Vertex -0.5,0.4,0.5
Vertex -1.5,0.4,0.5
Vertex -1.5,0,0.5

Face 1,6,7,0
Face 3,4,5,2
Face 9,14,15,8
Face 11,12,13,10
Color 200,200,200

[MeshBuilder]
Vertex -1.5,0,-0.5
Vertex -1.5,0.4,-0.5
Vertex -0.5,0.4,-0.5
Vertex -0.5,0.7,-0.5
Vertex 0.5,0.7,-0.5
Vertex 0.5,0.4,-0.5
Vertex 1.5,0.4,-0.5
Vertex 1.5,0,-0.5

Vertex -1.5,0,0.5
Vertex -1.5,0.4,0.5
```

```
Vertex -0.5,0.4,0.5
Vertex -0.5,0.7,0.5
Vertex 0.5,0.7,0.5
Vertex 0.5,0.4,0.5
Vertex 1.5,0.4,0.5
Vertex 1.5,0,0.5

Face 9,1,0,8
Face 10,2,1,9
Face 11,3,2,10
Face 12,4,3,11
Face 13,5,4,12
Face 14,6,5,13
Face 15,7,6,14
Face 0,7,15,8

Color 0,0,0
```

Hurrá, ezt kapjuk!



Elemezzük ki, mit és miért írtunk be! Vegyük először az első MeshBuilder részt!

Amint látjuk 2×8 Vertex parancsot írtunk be, amelyek között azért van hasonlóság. Az első nyolcas Z koordinátái negatívak, ezért az elülső felület(ek) az origó előtt lesznek. A második nyolcas Z koordinátái pozitívak, ezért az azok által meghatározott felületek az origó mögé esnek. Na de hát láthatjuk: az Y koordináták rendre megegyeznek, az X koordináták egymás ellentettjei. Ez azért van, mert a dobogó elülső és hátsó része ugyanaz (180 fokkal elforgatva). Az ellentettek pedig a már jól ismert „nézőpont probléma" miatt jelentek meg, azaz, hogy mindig az óramutató járásával megegyezően történjen a pontok összekötése. Remélem ez érthető, ha nem, akkor egy papíron kézzel megrajzolt ábra sokat segíthet. Így olyan példát is láthatunk, amikor 2 felületnek nincsenek közös pontjai, mégis létrehozhatók egy MeshBuilderben. Ezt tessék megjegyezni, 1-2 helyen még majd jól jöhet.

Okés, miért van mégis 4 (azaz 2×2) Face utasítás? A fent említett konvex-konkáv probléma miatt. A képen is látható, hogy az „első helyezett" magasabban áll. Így egy Face létrehozza a „2. és 3.

helyezett” szintjének megfelelő részt (alsó szürke sáv), a másik Face pedig az „első helyezett” részét (felső szürke sáv). Természetesen a hátsó oldalon is kettő Face végzi el a feladatot, ezért van összesen négy. Végül a felületek (elvileg ugye kettő, gyakorlatilag négy) szürke színt kaptak.

A második MeshBuilder szekció megértéséhez jól jön ugyancsak egy kézi rajz. Itt is 2x8 pont koordinátáit látjuk, ahol csak a Z paraméterben van eltérés. Ezt nem is taglalnám miért, hiszen a pontok szimmetrikusan vannak elhelyezve, csak abban térnek el, hogy elöl vannak vagy hátul. Összesen nyolc felületet hoztunk létre (az alapot is számold!) és feketére színeztük ezeket.

Remélem ez sem okozott nagy problémát megérteni, és ha ez így van, itt egy újabb feladatsor:

3. feladat

Ezek a feladatok szervesen felhasználják az előzőekben tanultakat.

a) Hozz létre egy kockát (rád bízom, milyen méretekkel, milyen pozícióba). Fesd ki az oldalait úgy, hogy a szemközt levő oldalai azonos színeket kapjanak! A színek kiválasztásában is egyénieskedhetsz.

Ugye sikerült mindössze három MeshBuilder szekcióval elkészíteni?!

b) Készíts el egy házat (lehet kocka vagy téglatest alakú, amelyiket akarod)! Fontos: legyen a háznak teteje (lehet sátortetős vagy gúla (=piramis) alakú). A méretekről, elhelyezésről, színekről most is te döntesz.

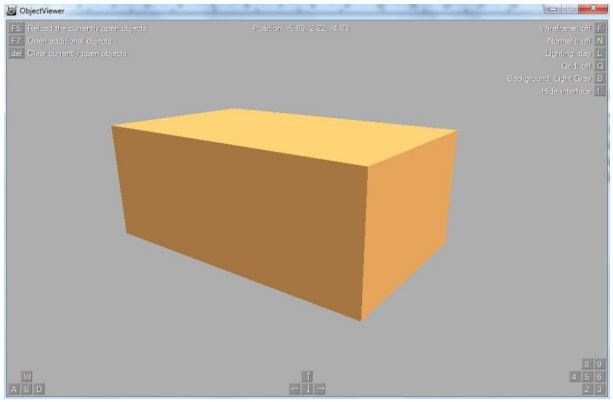
Primitívek

Primitíveknek az előre definiált, egyszerűbb objektumokat nevezzük. Ilyen például a hasáb (téglatest, kocka), gúla, csonkagúla, gömb, henger, tórus (gyűrű) stb… Aki már foglalkozott 3D-s modellező programmal, biztos találkozott velük. A BVE-ben kettő ilyen létezik, ez a Cube és a Cylinder. Az előbbivel téglatestet, kockát, négyzetet, téglalapot lehet létrehozni, míg az utóbbival kört, szabályos sokszöget, gúlát, csonkagúlát, kúpot, csonkakúpot, hengert. Miért jó ez? Például azért, ha egy sima egyszínű kockát akarunk létrehozni, akkor nem kell a 6 db felület pontjainak beírkálni, hanem csak a méreteit, és kész.

Lássuk akkor először a Cube parancsot!

Mint minden különálló részobjektum, ez is saját MeshBuilder szekciót igényel.

```
;Egyszerű téglatest Cube-bal
;Készítette: én
[MeshBuilder]
Cube 1.5,1,2.5
Color 255,178,96
```



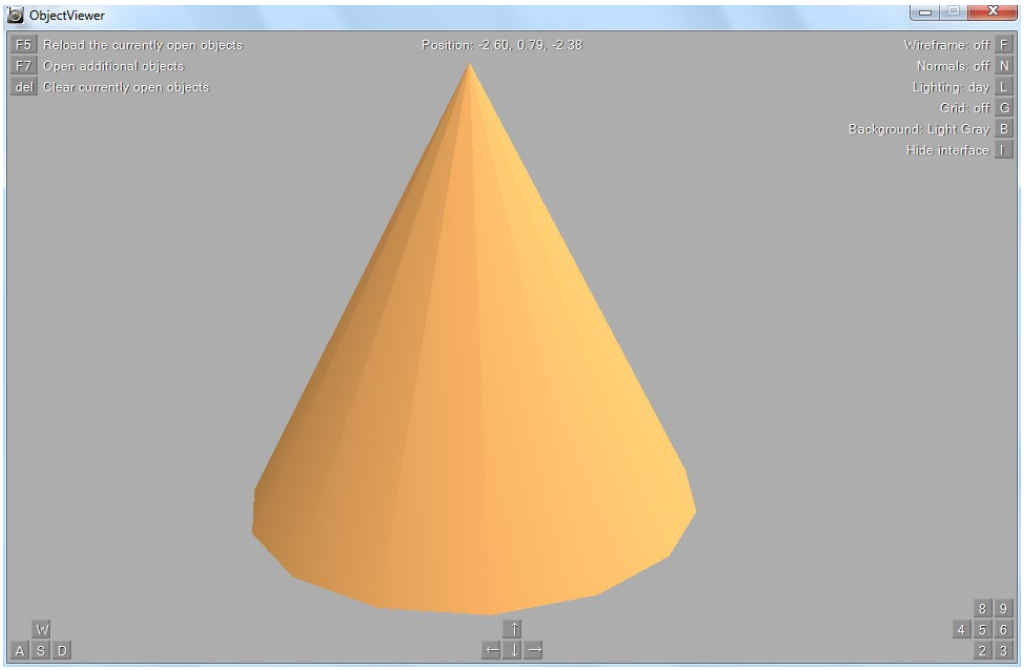
Ennyi. Láthatjuk, egy paranccsal létrejött egy téglatest, és ehhez csak a test méreteit kellett megadni (x, y és z irányban).

Fontos: a Cube a paraméterek kétszeresével hozza létre a testet, azaz a fenti paraméterekkel az ábrán látható téglatest 2×1.5=3 méter széles, 2×1=2 méter magas, és 2×2.5=5 méter mély.

Változtasd meg az értékeket és figyeld meg mi változik! Használj negatív értékeket vagy 0-t!

Na, akkor lássuk, mit tud a Cylinder!

```
;Cylinder
;Készítette: én
[MeshBuilder]
Cylinder 16,0,1.1,2
Color 255,178,96
```



Hát ez egy kúpnak tűnik. Tehát a Cylinder tökéletesen alkalmas kúp létrehozására. De mi történik, ha az első paraméterét pl. 4-re vagy 6-ra állítjuk?! Egyre kevésbé hasonlít kúpra. Ez azért van, mert ez eredendően egy gúla létrehozására szolgál, olyanéra, amelynek alapja egy szabályos *n* oldalú sokszög. Mi éppen ezt az n-t állítottuk át. Tehát ha az első paraméter kicsi, akkor torzabb

a kúp, ha nagy, akkor az alap egyre jobban hasonlít egy körre, tehát egészében a test egy kúpra. Vigyázni kell vele, mert a nagy n a megjelenítés sebességének rovására megy! A második paraméter legyen akkor most pl. 0.5. Hát ez bizony egy csonkakúp/csonkagúla. Tehát a második paraméter a felső alap sugarát adja meg, ebből következik, hogy a harmadik paraméter viszont az alsóét.

Ha a két alap sugara megegyezik, hengert kapunk. Végül a negyedik paraméter a test hosszúságának (!) beállítására szolgál. Jól jegyezd meg: míg a Cube esetén a méretek felét kell megadni, addig itt a test valós hosszúságát! Próbáld ki mi történik, ha változtatgatod az értékeket. Tudom, sokadszor említem ezt a gyagyás „próbáld ki"-t, de hidd el, csak így lehet megérteni, mi mit csinál.

Gondolom észrevetted, hogy bárhogy is állítgatod a fenti két primitív paramétereit, sosem fog pl. ferdén állni, nem lesz elforgatva stb. Ehhez tudni kell azt, hogy a BVE a primitíveket origóközpontúan hozza létre, azaz pl. a kocka vagy a henger közepe pont az origóban lesz. „Jó,jó, de ez így nem sok" – mondhatod, és igazad van. Hiszen ha több részobjektumot az origóba hozol létre, akkor az egész zagyvaság lesz. Nem kell megijedni, a BVE támogatja a fenti problémák megoldását, azaz át lehet ezeket helyezni, sőt forgatni is.

Áthelyezés, forgatás

A Koordinátarendszer fejezetnél említettem milyen fontos, hogy egy tárgy origókezdetű vagy origóközpontú, ugyanis a program áthelyezésnél a primitívek középpontját helyezi át a megadott pontba, így a többi pontjait is ehhez viszonyítja. Sőt, nem csak primitíveket, de egy MeshBuilder szekció összes felületét (Face) is át lehet helyezni egy adott pontba. Fontos, hogy ilyenkor mintha az origót helyezné át a program a megadott pontba, és ez a pont lesz az „új" origó. Például ha van egy felületünk, ennek egy pontja (2,-1, 2), és az egészet áthelyezzük a (3,3,3)-ba akkor az előző pont koordinátái: (5, 2, 5) lesznek.

A forgatás pedig az origó körül zajlik, ezért ahogy fentebb említettem, így az origóközpontú objektumokat sokkal könnyebb forgatni abból a szempontból, hogy nem kell azzal számolni, melyik pont hova kerül majd.

Ezek után nézzük a konkrét parancsokat!

Áthelyezés

Az áthelyezés parancsa a **Translate**:

```
[MeshBuilder]
Cylinder 8,0.5,0.5,2
Color 255,178,96
Translate 1,0,0
```

Ez áthelyezi a felület/primitív középpontját az (1,0,0) pontba! Változtatgasd az értékeket és figyeld meg mi történik!

Létezik még egy parancs az áthelyezéshez, méghozzá a **TranslateAll X,Y,Z**. Ez csak OpenBVE-ben használható, BVE2-ben és BVE4-ben nem. Ez a parancs ugyanúgy működik, mint a sima Translate, viszont nem csak annak a MeshBuilder szekciónak a felületeit helyezi át, amelynek a végén a parancs áll, hanem az összes megelőző MeshBuilderét is. Ha a B3D fájl legvégére, utolsó sorként írjuk be ezt a parancsot, akkor az egész objektumot áthelyezi.

Forgatás

A forgatás parancsa a **Rotate X,Y,Z,fok**.

```
[MeshBuilder]
Cylinder 8,0.5,0.5,2
Color 160,160,160
Rotate 1,0,0,30
```

A fenti részlet a hengert elforgatja 30 fokkal az YZ síkban. Tehát, ha az x paraméternél áll szám, akkor az x-tengely körül, ha az y paraméter van beállítva akkor az y-tengely körül, míg a 3. paraméter esetén a z-tengely körül fordul el az objektum/felület a megadott szöggel (fok). Ha a szög megadásánál negatív számot használsz, akkor ellenkező irányú forgatás történik. Természetesen lehet két forgatást egybe is kombinálni, azaz lehet egyszerre az x és y tengely körül forgatni ugyanazzal a szöggel. Sőt a parancsok egymás után is használhatóak. Célszerű ilyen esetben először forgatni a megfelelő szöggel, és utána áthelyezni.

Létezik még egy parancs a forgatáshoz, méghozzá a **RotateAll X,Y,Z,fok**. Ez csak OpenBVE-ben használható, BVE2-ben és BVE4-ben nem**.** Ez a parancs ugyanúgy működik, mint a sima Rotate, viszont nem csak annak a MeshBuilder szekciónak a felületeit forgatja el, amelynek a végén a parancs áll, hanem az összes megelőző MeshBuilderét is. Ha a B3D fájl legvégére, utolsó sorként írjuk be ezt a parancsot, akkor az egész objektumot elforgatja.

Tessék megint változtatgatni az értékeken! Ha érthető minden, akkor itt az újabb feladat:

4. feladat

Készíts el egy háromfokú létrát! A létra két „szára” legyen téglatest, míg a fokai hengerek. A méretek és szín kiválasztását újfent rád bízom.

Kész!! – Hogyan tovább?

Ha a fenti feladatokat sikerrel tudtad teljesíteni, megértetted a parancsokat, akkor örömmel kijelentheted, „tudok modellezni”. Tudod mi az a test, felület, pontok. Tudod hogyan kell a térben pontokat definiálni, ezekből felületeket létrehozni. Tudod hogyan kell a felületeket komplexebb testekké összekapcsolni. Tisztában vagy a primitívek fogalmával, létrehozásuk módjával. Tudod mi az az áthelyezés, forgatás, érted ezen parancsok működését. A színekkel sem állsz hadilábon, tudod hogyan kell felületeket, testeket kiszínezni. Így összességében létre tudsz hozni már bármilyen komplexitású tárgyat – természetesen csak tőled függ, milyen részletességgel dolgozol ki valamit. Ahhoz, hogy teljes legyen a kép, már csak egy témakörről nem említettünk szót: ez a

textúrák használata. Mielőtt ebbe belemennénk, essen pár szó a poligonszámról is!

Első szempont: ahogy már említettem, csak rajtad múlik, milyen részletességgel kívánsz kidolgozni egy objektumot, illetve annak egy-egy részét. A túl kevés felület (poligon), részobjektum nem túl esztétikus – bár a síntől távolabbi helyeken a célnak megfelelhet. A túl sok poligonnal rendelkező, túl részletes objektum viszont már nagyon befolyásolja a megjelenítés sebességét. Főleg akkor, ha egy kisebb körzetbe helyezel el többet belőlük. Éppen ezért a sínektől távoli fáknak, bokroknak éppen csak annyira szükséges kidolgozottnak lennie, amennyiből jól felismerhetőek lesznek, Viszont a sínekhez közeli tárgyakkal, kocsikkal, járművekkel, épületekkel szemben elvárható a megfelelő szintű kidolgozottság. Meg kell találni a kettő között valahol az arany középutat és sajnos igen, kompromisszumokat kell kötni.

Textúrák

Mi az a textúra, miért van rá szükség? Az első kérdésre a válasz: a textúra tulajdonképpen nem más, mint egy kép vagy annak részlete, amely az objektumunk valamelyik felületén fog díszelegni (általában fényképeket takar a megnevezés) A második kérdésre válaszolva, tulajdonképpen nincs rájuk szükség, hiszen láthattuk: sima Color utasításokkal is el tudjuk érni a kívánt hatást (többé-kevésbé). Akik ezt nem így gondolják, azok számára ez a fejezet igen hasznos lesz.

Tehát akkor még egyszer: a textúra egy kép. Ezt a képet PNG vagy BMP fájlok tárolják. A PNG csak OpenBVE-ben használható, a BMP OpenBVE-ben és BVE2, BVE4-ben is. Tehát ha azt szeretnéd, hogy az objektum a BVE régi változataiban is működjön, akkor BMP textúrákat használj. Ha ez nem fontos számodra, akkor a PNG az ajánlott. A legalapvetőbb program a textúrák szerkesztésére a Windowsba beépített Paint program, ez egy darabig megteszi, de idővel érdemes lesz megismerkedni a Paint.NET-tel, később, jó adag tapasztalatszerzés után esetleg a GIMP-pel is.

A formátumról röviden: a kép hosszának és magasságának (width/height) a 2 nemnegatív egész számú hatványának kell lennie, azaz a következő méretek vannak megengedve: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, és így tovább. Kis méretű tárgyakra elég kisebb méretű textúra, nagyobb méretű tárgyakra, például egy vasúti kocsira 1024 képpontnál keskenyebb textúra nem ajánlott. Más 3D-s programoktól eltérően a képnek nem szükséges négyzet alakúnak lennie, azaz a hossznak és a magasságnak nem feltétlenül kell megegyeznie.

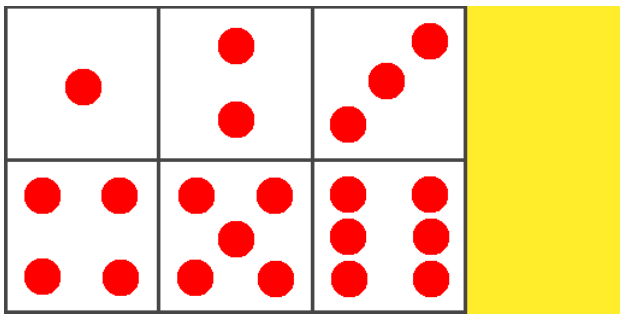
A képek lehetőleg valódi fényképek legyenek, ne kézi rajzok.

Mekkora részletet tartalmaz egy PNG vagy BMP fájl? Ez tőled függ és magától a fájltól. Egy PNG vagy BMP fájlban tárolhatsz több részobjektumhoz (felülethez) szükséges textúrarészt, de azt az elvet is követheted, hogy egy részobjektum – egy képfájl.

Mit is tudunk textúrázni? Mindent, ami egy MeshBuilder szekcióban szerepel. Ebből következik, hogy a textúrázást definiáló rész csak a MeshBuilder rész után következik, az ott definiált tárgyak/felületek textúráiról hordoz információt. „Legkönnyebb” a sima Face-szel létrehozott felületekre textúrát húzni. A primitíveket is el lehet látni képpel, csak az eredmény nem olyan

lesz, amilyent várnánk (legalábbis a Cube esetében biztos nem) – ennek oka, hogy a Cube/Cylinder objektumok csúcsai ugyanúgy be vannak számozva, viszont ezek összekötögetése miatt speciális textúrát igényelnek (főleg a Cube). Itt ezek textúrázását nem részletezném, a Cylinderről annyit hogy a felső szabályos sokszög csúcsainak számozása: 2n, az alsóé 2n+1, ahol n=0,1,2,... n-1. Az alkotó megjelenítése még csak-csak elmegy, de az alapoké már kevésbé. A Cube textúrázása már bonyolultabb, kísérletezgetni kell vele (megjegyzem nekem még nem sikerült egy téglatestet se rendesen ellátni képpel, ilyenkor ezeket inkább 6 db Face-szel adom meg, téglalap alakú felület textúrázása gyerekjáték).

Itt is nézzünk meg egy példát: csináljunk egy dobókockát. A textúrát Paintben fogom megrajzolni: 512×256-os méretet választok, 6 db négyzetet rajzolok, 3-3-at egymás mellé egy-egy sorba, illetve belerajzolom a pöttyöket. Az egész valahogy így néz ki. A sárga részt nem fogjuk felhasználni a képből.



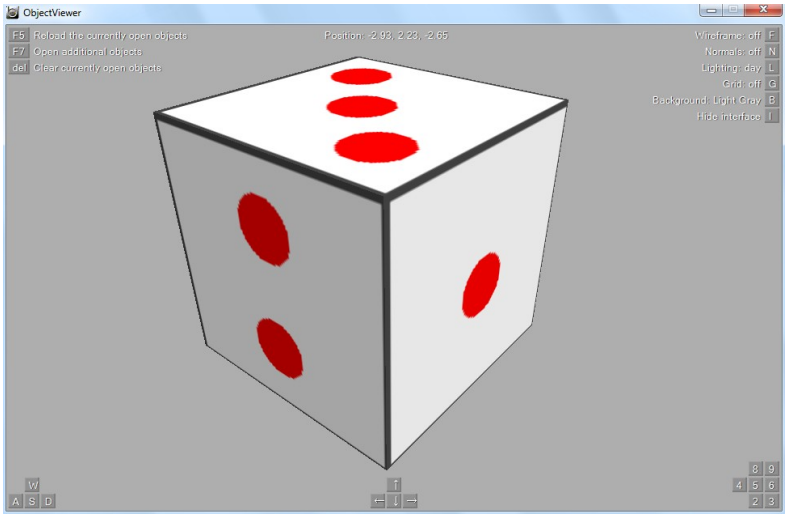
Elmentettem kocka.png néven, PNG fájlként. Ezenkívül feljegyzem a négyzetek határpontjait. A bal felső sarok a (0,0), a jobb alsó a (384,256). A felső és az alsó sor közötti vonal pont a kép felénél van, tehát a kép tetejétől mérve ez 256/2=128. A harmadoló vonalak: 384/3=128, illetve 256. Egyelőre ennyi.

A kocka elkészítése a szokásos módon zajlik, különbség, hogy most már textúrákat is használunk. Fentebb láthattuk, hogy a kocka két szembelevő oldalát közös MeshBuilder részben is létrehozhatjuk, de a textúrázás megértése miatt az oldalak definíciója külön-külön történik. Akkor a dobókocka B3D fájlja:

```
;Dobókocka textúrákkal
;Készítette: én

;1-es
[MeshBuilder]
Vertex -1,1,-1
Vertex 1,1,-1
Vertex 1,-1,-1
Vertex -1,-1,-1
Face 0,1,2,3
Load kocka.png
Coordinates 0,0,0
Coordinates 1,0.25,0
Coordinates 2,0.25,0.5
Coordinates 3,0,0.5
```

```
;2-es
[MeshBuilder]
Vertex -1,1,1
Vertex -1,1,-1
Vertex -1,-1,-1
Vertex -1,-1,1
Face 0,1,2,3
Load kocka.png
Coordinates 0,0.25,0
Coordinates 1,0.5,0
Coordinates 2,0.5,0.5
Coordinates 3,0.25,0.5

;3-as
[MeshBuilder]
Vertex -1,1,1
Vertex 1,1,1
Vertex 1,1,-1
Vertex -1,1,-1
Face 0,1,2,3
Load kocka.png
Coordinates 0,0.5,0
Coordinates 1,0.75,0
Coordinates 2,0.75,0.5
Coordinates 3,0.5,0.5

;4-es
[MeshBuilder]
Vertex -1,-1,-1
Vertex 1,-1,-1
Vertex 1,-1,1
Vertex -1,-1,1
Face 0,1,2,3
Load kocka.png
Coordinates 0,0,0.5
Coordinates 1,0.25,0.5
Coordinates 2,0.25,1
Coordinates 3,0,1

;5-ös
[MeshBuilder]
Vertex 1,1,-1
Vertex 1,1,1
Vertex 1,-1,1
Vertex 1,-1,-1
Face 0,1,2,3
Load kocka.png
```

```
Coordinates 0,0.25,0.5
Coordinates 1,0.5,0.5
Coordinates 2,0.5,1
Coordinates 3,0.25,1

;6-os
[MeshBuilder]
Vertex 1,1,1
Vertex -1,1,1
Vertex -1,-1,1
Vertex 1,-1,1
Face 0,1,2,3
Load kocka.png
Coordinates 0,0.5,0.5
Coordinates 1,0.75,0.5
Coordinates 2,0.75,1
Coordinates 3,0.5,1
```



Ennyi is lenne, akkor elemezzük ki a textúrázást!

Ahogy említettem, minden MeshBuilder szekció után jöhet a textúra meghatározása. Láthatjuk, hogy nem használtunk Color utasítást. Nem is lenne értelme, mivel most nem kell egy adott színre kiszínezni a felületet, hanem külön képet rendelünk hozzá. A textúránk ezúttal nem fénykép, de jelenleg ez is megteszi.

A textúráról szóló szakasz első sora a Load parancs. A Load után a használandó képfájl nevét kell megadni. Célszer a képfájlt ugyanabba a könyvtárba elhelyezni, amelyikben a modellfájlt. Egyéb más esetben a Load után a relatív elérési útvonallal kell kiegészíteni a PNG vagy BMP fájlnevet (például: ..\kepek\kocka.png).

Ha BVE2-ben és BVE4-ben is működőképes modellt szeretnél készíteni, akkor a Load parancs elé, külön sorba be kell írni azt, hogy *[Texture]*, ez a textúráról szóló szakasz bevezetője. Ez OpenBVE-hez nem szükséges, így az előbbi B3D fájlban ezt kihagytuk. Nézzük meg, hogy nézne ki az előbbi B3D fájl első MeshBuilder szekciója BVE2 és BVE4 kompatibilisen:

```
;Dobókocka textúrákkal
;Készítette: én

;1-es
[MeshBuilder]
Vertex -1,1,-1
Vertex 1,1,-1
Vertex 1,-1,-1
Vertex -1,-1,-1
Face 0,1,2,3
[Texture]
Load kocka.png
Coordinates 0,0,0
Coordinates 1,0.25,0
Coordinates 2,0.25,0.5
Coordinates 3,0,0.5
```

Láthatjuk, a Face parancs után, a Load parancs előtt ott a [Texture]. Mivel manapság általában már nem számít a BVE2 és BVE4 kompatibilitás, mi a [Texture] sort ki fogjuk hagyni a példáinkban.

A Load parancsot követő négy sorban 4 db Coordinates parancsot látunk, pontosan annyit, amennyi pontból az adott felületünk áll. Így kitalálhatjuk: a Coordinates utasítás első paramétere azt a pontot határozza meg, amelynek a (textúrabeli) koordinátáit akarjuk megadni.

Hogyan számíthatóak ki ezek a koordináták? A Coordinates parancs második paramétere azt adja meg, hogy vízszintesen hányszorosára húzza rá a program a textúrát az adott pontra; a harmadik paraméter pedig a függőleges irányt. Lássuk a kockán az '1-es' textúrázását! Nézzük a fenti PNG-n (amelyiken a 6 négyzet szerepel) az 1-est! A bal felső sarok a (0,0)-ban található, a jobb felső sarok a (128,0)-ban, a jobb alsó (128,128), bal alsó (0,128). Ezt a Paintben megállapíthatod, ha a metszéspontra rámész az egérmutatóval, akkor a program az alsó szürke sávban jobb oldalt kiírja a fenti értékeket. Már csak a kép méretére van szükség, de ezt tudjuk, 256×128 (így hoztuk létre). A Coordinates 0 a bal felső sarkot írja le: ez 0/512=0 és 0/256=0, ezért a további paraméterek: 0,0. A Coordinates 1 a jobb felső sarokra vonatkozik: 128/512=0.25 és 0/256=0, így ide 0.25,0-t írunk. A Coordinates 2 a jobb alsó sarkot jelenti: 128/512=0.25 és 128/256=0.5, ez 0.25,0.5.

Végül hasonló módon számolva: Coordinates 3,0,0.5

Így egy négyzetet „húztunk" egy négyzetre, ezért látjuk azt az egyetlen piros pöttyöt. Ha például a 1-es és a 2-es pont textúrakoordinátáinak a 0.5,0 illetve 0.5,0.5-t adtuk volna meg, akkor 1 négyzeten belül összesen 3 pöttyöt láthatnánk (az 1-est és a 2-est). Próba!

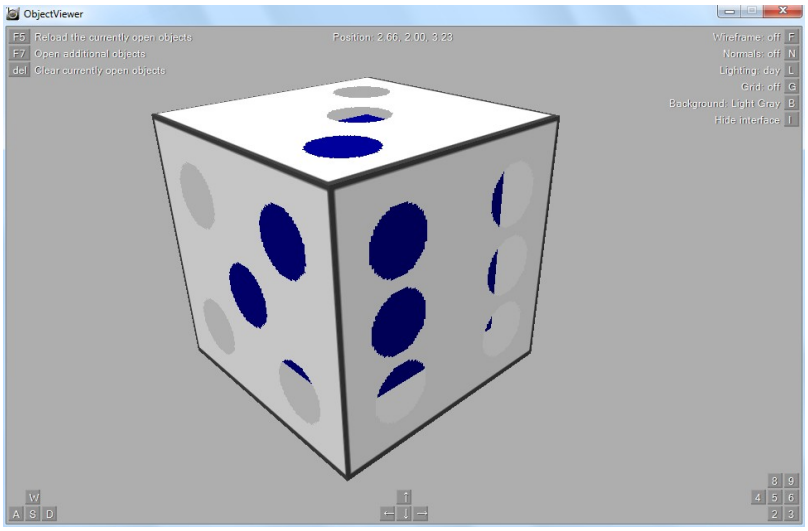
A többi oldal textúrázása ugyanígy megy, feladatként ezt próbáld meg átgondolni.

Természetesen negatív számokat is használhatunk, akkor az ellenkező irányban (tk. a tükörképét) húzza rá a textúrát a pontra, illetve lehetnek 1-nél nagyobbak is, akkor a kép többszörösét jeleníti meg a program. Ezt is próbáld ki!

Arra is szükségünk lehet, hogy az objektumok bizonyos része átlátszó legyen (pl. ablakok, alagút stb.). Ilyen esetben a PNG vagy BMP fájlok ezen részeit át kell festeni egy adott színre, majd a szín RGB összetevőjét megjegyezni (a már ismert módon: Paintben rákattintunk a kívánt színre, Színek/Színek szerkesztése… stb.).

Legyen most a piros szín az átlátszó, más szóval transzparens szín: azaz azok a részek, amelyek piros színnel (RGB=255,0,0) vannak kifestve, átlátszóak lesznek. Írjuk be most a Texture szekciókba, minden egyes utolsó Coordinates utasítás után, hogy: *Transparent 255,0,0*!

Nézzük meg az eredményt Object Viewerben!



Megjegyzés: kiegészítettem a fájlt a végén egy rövid kóddal, amely egy kék kockát hoz létre a nagy kocka belsejében.

```
;Kis kocka
[MeshBuilder]
Cube 0.75,0.75,0.75
Color 0,0,128
```

Láthatjuk, hogy azokon a helyeken, ahol a PNG piros színnel volt festve (pöttyök) most átlátszóak lettek, rajtuk keresztül előtűnik a kis kék kocka.

Éjszakai textúrák

Ha végiggondoljuk mindazt amit eddig olvastunk, lassan azon is elkezdünk gondolkodni, hogy amikor már gyakorlottak leszünk a modellezésben melyik az a kedvenc mozdonyunk, villamosunk, metrószerelvényünk, amit majd egyszer szeretnénk megépíteni. Elképzeljük az épülő kocsiszekrényt, a homlokfalat, a vezetőállást. Hoppá, a vezetőállás! Itt lesz egy kis gond! A vezetőállás egész máshogy néz ki nappal, mint éjjel. Nappal világos, tisztán látható, éjjel viszont sötét, a kijelzők kis lámpái viszont világítanak. Szóval tulajdonképpen ide nem is egy, hanem kétféle textúra volna szükséges, és ez lehetséges is. Nézzük meg ezt a MeshBuildert:

```
[MeshBuilder]
Vertex 0,2.2,7
```

```
Vertex 1.4,2.2,7
Vertex 1.4,2,6.3
Vertex 0,2,6.3
Face 0,1,2,3
Load asztal-nappal.png,asztal-ejjel.png
Coordinates 0,0,0
Coordinates 1,1,0
Coordinates 2,1,1
Coordinates 3,0,1
```

Nézzük meg jól a Load parancsot. Nem egy kép neve áll ott, hanem kettő, vesszővel elválasztva. Igen, a második az éjszakai textúra lesz. Ezzel megoldhatjuk az előbb leírt problémát, nappali fényben az asztal-nappal.png textúra fog megjelenni, sötétben pedig az asztal-ejjel.png. Erre csak OpenBVE-ben van lehetőség, BVE2-ben és BVE4-ben nincsen külön éjszakai textúra.

Fontos, hogy a két textúra a fényviszonyoktól függően „keveredhet” is. Ez azt jelenti, hogy például szürkületben valamennyire még látszik a nappali textúra, de már az éjjeli is megjelenik. Az OpenBVE ügyesen tudja keverni a kettőt. Emiatt a kevert megjelenítés miatt ügyeljünk arra, hogy a nappali és az éjszakai textúrakép ugyanabból a nézőpontból készüljön és a két kép fedje egymást, vagyis a kijelzők, kapcsolók és más alkatrészek mindkét képen ugyanazon a helyen legyenek.

Felmerülhet a kérdés, hogy akkor most minden felületre kell külön nappali és éjszakai textúra? A válasz szerencsére *nem*. Általában elég csak nappali textúrát használni, ezt az OpenBVE automatikusan fogja sötétíteni a fényviszonyoknak megfelelően. Olyan tárgyaknál célszerű használni éjszakai textúrát, amelyek éjjel másképpen jelennek meg. Ilyen az említett vezetőállás a mozdonyokon, villamosokon, metrókon, de esetleg ilyen lehet a pályához közeli házak ablaka is.

Lámpák, jelzők, világító tárgyak

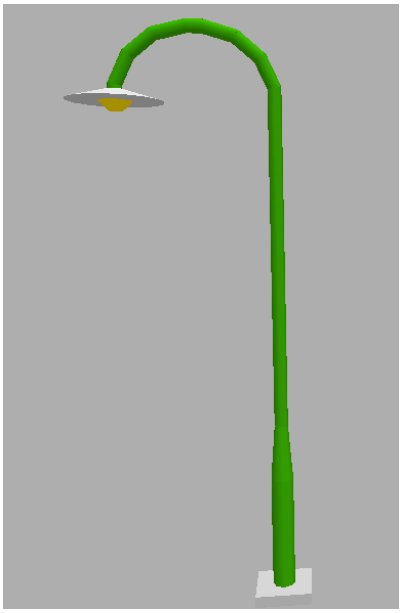
A pályaépítők gyakran készítenek éjszakai változatot is a pályáikból, és éjszaka szükség van világításra. A peronon levő lámpák, a vasúti jelzők, a közúti forgalomirányító lámpák mind-mind fényt bocsájtanak ki magukból. Ezt OpenBVE-ben is megvalósítjuk (BVE2-ben és BVE4-ben viszont nem). Készítettem egy sematikus lámpaoszlopot az alábbi B3D fájllal:

```
;Alapzat
[MeshBuilder]
Cube 0.25,0.05,0.25
Color 180,180,180
Translate 0,0.05,0

;Oszlop alsó része
[MeshBuilder]
Cylinder 8,0.1,0.1,1
Color 48,160,0
Translate 0,0.6,0
```

```
;Oszlop vékonyodó része
[MeshBuilder]
Cylinder 8,0.06,0.1,0.5
Color 48,160,0
Translate 0,1.35,0

;Oszlop vékony felső része
[MeshBuilder]
Cylinder 8,0.06,0.06,3
Color 48,160,0
Translate 0,3.1,0

;Felső hajlított rész - első darab
[MeshBuilder]
Cylinder 8,0.06,0.06,0.3
Color 48,160,0
Rotate 0,0,1,25
Translate -0.058,4.71,0

;Felső hajlított rész - második darab
[MeshBuilder]
Cylinder 8,0.06,0.06,0.3
Color 48,160,0
Rotate 0,0,1,50
Translate -0.22,4.92,0

;Felső hajlított rész - harmadik darab
[MeshBuilder]
Cylinder 8,0.06,0.06,0.3
Color 48,160,0
Rotate 0,0,1,75
Translate -0.455,5.043,0

;Felső hajlított rész - negyedik darab
[MeshBuilder]
Cylinder 8,0.06,0.06,0.15
Color 48,160,0
Rotate 0,0,1,90
Translate -0.658,5.08,0

;Felső hajlított rész - ötödik darab
[MeshBuilder]
Cylinder 8,0.06,0.06,0.3
Color 48,160,0
Rotate 0,0,1,-75
Translate -0.861,5.043,0

;Felső hajlított rész - hatodik darab
```

```
[MeshBuilder]
Cylinder 8,0.06,0.06,0.3
Color 48,160,0
Rotate 0,0,1,-50
Translate -1.096,4.92,0

;Felső hajlított rész - hetedik darab
[MeshBuilder]
Cylinder 8,0.06,0.06,0.3
Color 48,160,0
Rotate 0,0,1,-25
Translate -1.258,4.71,0

;Oszlop vékony felső része
[MeshBuilder]
Cylinder 8,0.06,0.06,0.1
Color 48,160,0
Translate -1.316,4.55,0

;Lámpabúra
[MeshBuilder]
Cylinder 16,0.06,0.4,0.1
Color 190,190,190
Translate -1.316,4.5,0

;Izzó
[MeshBuilder]
Cylinder 16,0.14,0.06,0.08
Color 255,216,0
Translate -1.316,4.41,0
```

Így néz ki Object Viewerben:



Ez rendben is van, de így még nem világít. Most hozzáadjuk a világításhoz szükséges **EmissiveColor R,G,B** parancsot az utolsó MeshBuilderhez, vagyis az izzóhoz:

```
;Izzó
[MeshBuilder]
Cylinder 16,0.14,0.06,0.08
Color 255,216,0
Translate -1.316,4.41,0
EmissiveColor 255,216,0
```

Záró feladat

Ennek a feladatnak már nem fogod a megoldását megtalálni ebben a leírásban. Most már tényleg mindent tudsz a BVE objektumok modellezéséről (kivéve a primitívek textúrázását – bár ez legyen a legnagyobb gondod). Ha végigcsináltad az összes feladatot és nem okoztak nagy problémát, akkor kijelentheted magadról: bármilyen komplexitású objektumot el tudsz készíteni. Zárófeladatnak arra gondoltam, hogy nézz ki magadnak egy tárgyat/épületet/járművet, stb. a környezetedből, készíts róla fényképeket vagy tölts le róla képeket internetről (ha találsz), végső esetben Painttel rajzolj. Ha kisebb a tárgy (pl. könyv) akkor mérőszalaggal lemérheted a dimenzióit, ha nagyobb akkor megsaccolod. Az adatok alapján készítsd el a modellt BVE-hez és rakj rá textúrát!

Tanács: nem kell egyből a legbonyolultabbal kezdeni, javaslataim: könyv, telefonfülke, panelház, televízió, stb. Ezek mind-mind nagyon egyszerű objektumok, de nem baj: a következő munkád már olyan legyen, amelyben ívek vannak (pl. gépkocsi), utána még komplexebb és így tovább. A lényeg: tartsd szem előtt a *Hogyan tovább* részben leírtakat.

Zárszó

Itt a vége, fuss el véle! Remélem érthető és hasznos volt ez a leírás. Azoknak is akik még csak most szeretnék megtanulni a BVE objektumok modellezését, de azoknak is, akik már elkészítettek 1-2 tereptárgyat. Azt is remélem, hogy ezen doksi nem rettentett el senkit az objektumkészítéstől és azért ha nem is több száz embert, de 1-2 „ojjektumkészítő jómunkásembert” köszönthetünk a magyar BVE fejlesztői táborban.

Én részemről úgy vélem, hogy aki rendesen végigcsinálta mindazt, ami a leírásban található, annak tényleg nem okozhat gondot egy komplexebb objektum elkészítése, bár ehhez tudomásul kell venni azt, hogy 1-1 mozdony legyártása nem megy az egyik napról a másikra. Én sem a Szilivel kezdtem, sőt volt, hogy a Szilit félkész állapotban töröltem és újrakezdtem több órás munkát és még most is tudnék javítani rajta. Ahogy a Záró feladatnál említettem, egy egyszerű tárgy modellezésével kell kezdeni és onnan kell továbblépni. Igen, ott kell ülni a számítógép mellett, számolni, gépelni, javítani, újrakezdeni. Úgy senki nem fog megtanulni modellezni BVE-hez Jegyzettömbbel, hogy a leírt feladatokat nem csinálja meg, nem változtatgatja a paramétereket, nem foglalkozik a dologgal. Aki ezt nem veszi tudomásul, az ne sírjon, hogy a modellezés túl bonyolult és nem érti. Ha valaki mégis ezek ellenére elakadt (ismétlem: rendesen végigcsinált mindent) és nem érti, mit és hol rontott el, kérdezzen az alábbi címen vagy az

OpenBVE Trains Facebook csoportban. Ígérem, értelmes kérdésekre válaszolok.

Sok sikert kívánok, remélem segíthettem!

Krisz

Copyright és társai

Az itt közölt modellfájl-részletek a sajátjaid. Semmi különöset nem tartalmaznak, javítsd ki, módosítsd, terjeszd ha úgy tarja kedved. Ez a fájl is terjeszthető, de nem módosítható. Ha kérdésed, javaslatod van a leírással kapcsolatban, itt elérhetsz:

ckirbi@freemail.hu

Megoldások

1. feladat

a pont helye

a, a talaj alatt, tőlünk jobbra, előre

b, a talajjal egy szintben, tőlünk balra, előttünk

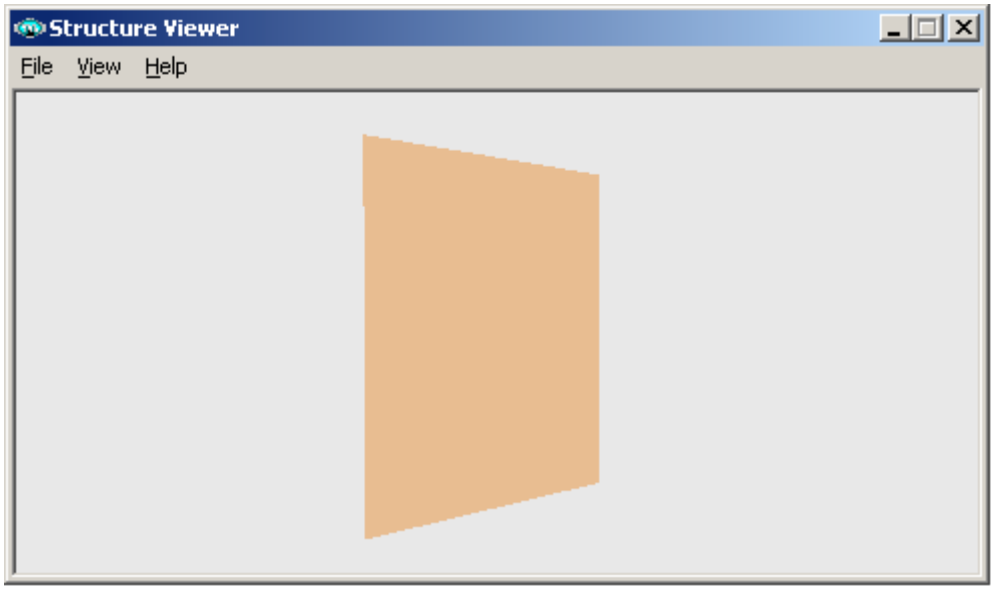
c, tőlünk jobbra, a föld fölött, a hátunk mögött

d, az origóval egyvonalban, de a föld fölött 1.5 magasan van (úgy is mondhatnák ez a szívünk egy kb 1.7 m magas ember esetében)

2. feladat

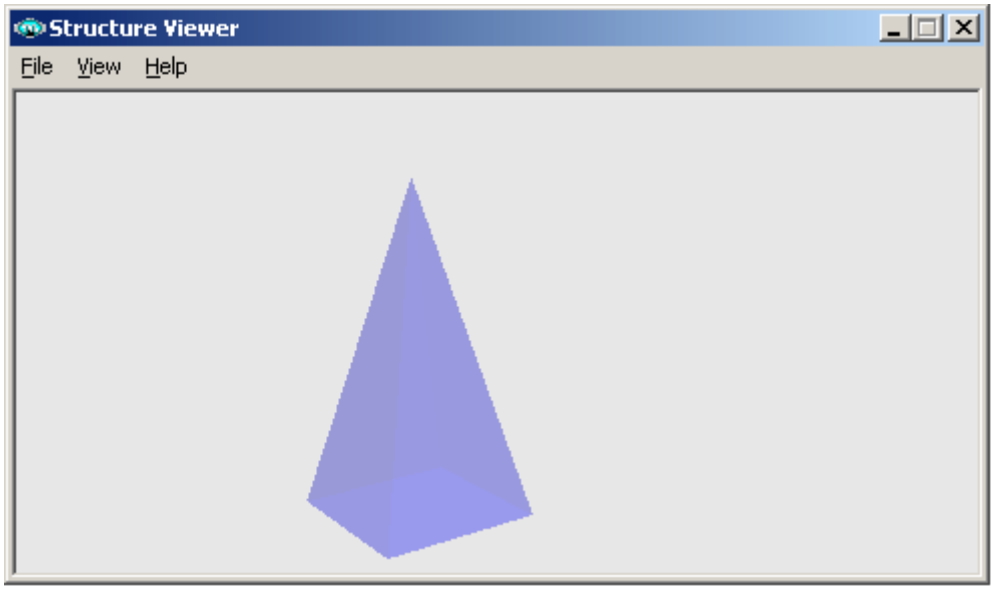
a)

```
[MeshBuilder]
Vertex 0,1.25,-1.25 ; 2.5 méter hosszú, tehát az origótól csak a fele!!
Vertex 0,1.25,1.25
Vertex 0,-1.25,1.25
Vertex 0,-1.25,-1.25
Face2 0,1,2,3
Color 255,128,0,100 ; narancs szín összetevői + az átlátszóság
```



b)

```
[MeshBuilder]
Vertex -0.75,0,-0.75 ;az alap pontjai – ezek közösek
Vertex 0.75,0,-0.75
Vertex 0.75,0,0.75
Vertex -0.75,0,0.75
Vertex 0,3,0 ;vigyázat! A csúcsot csak 1x kell létrehozni
Face2 0,1,2,3 ; ez itt az alap
Face2 0,1,4 ; a következő 4 Face2 a 4 db háromszöget hozza létre
Face2 1,2,4
Face2 2,3,4
Face2 3,0,4
Color 0,0,255,50 ; kékre színezünk
```

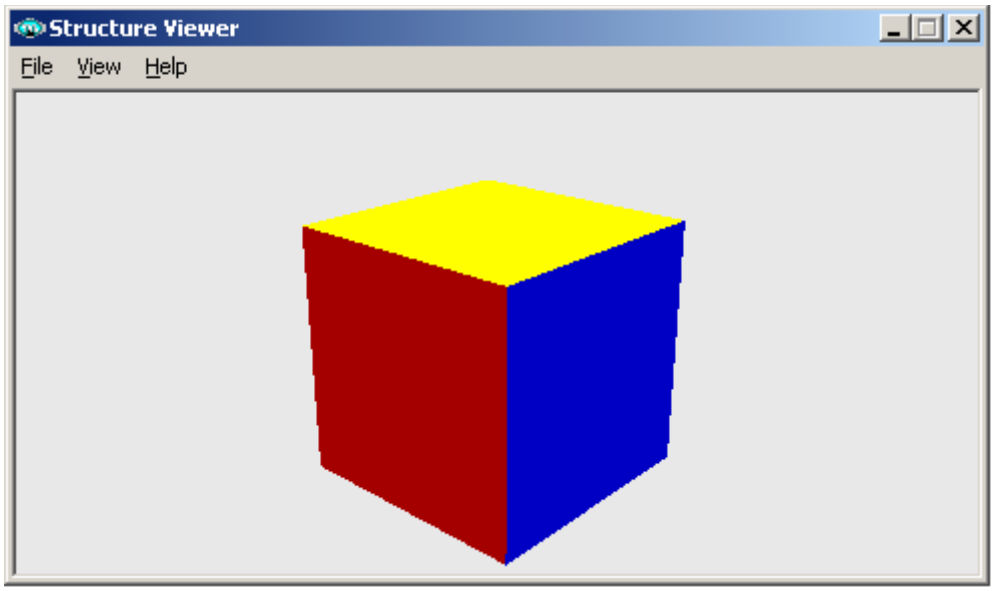


3. feladat

a)

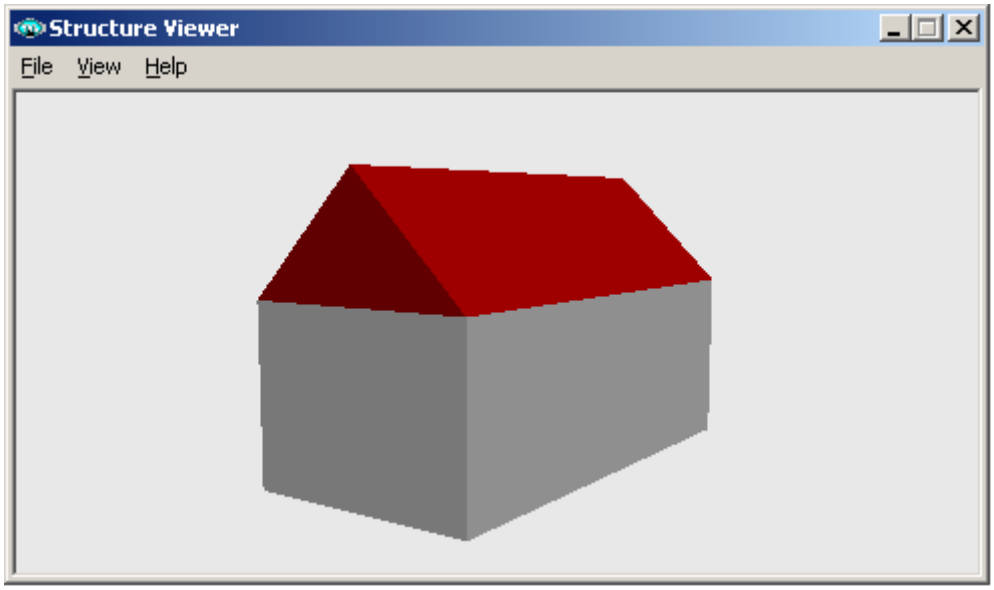
```
[MeshBuilder] ;2x2x2 méteres kocka
Vertex -1,1,-1
Vertex 1,1,-1
Vertex 1,-1,-1
Vertex -1,-1,-1
Vertex 1,1,1
Vertex -1,1,1
Vertex -1,-1,1
Vertex 1,-1,1
Face 0,1,2,3
Face 4,5,6,7
Color 255,0,0
```

```
[MeshBuilder]
Vertex -1,1,1
Vertex -1,1,-1
Vertex -1,-1,-1
Vertex -1,-1,1

Vertex 1,1,-1
Vertex 1,1,1
Vertex 1,-1,1
Vertex 1,-1,-1
Face 0,1,2,3
Face 4,5,6,7
Color 0,0,255

[MeshBuilder]
Vertex -1,1,1
Vertex 1,1,1
Vertex 1,1,-1
Vertex -1,1,-1
Vertex -1,-1,-1
Vertex 1,-1,-1
Vertex 1,-1,1
Vertex -1,-1,1
Face 0,1,2,3
Face 4,5,6,7
Color 255,255,0
```



b) (Megjegyzés: A ház alját „elhanyagoltam”)

```
[MeshBuilder] ;a ház téglatest alakú lesz
Vertex -2,0,4
Vertex 2,0,4
Vertex 2,0,-4
Vertex -2,0,-4
Vertex -2,3,4
Vertex 2,3,4
Vertex 2,3,-4
Vertex -2,3,-4

Face 4,7,3,0
Face 7,6,2,3
Face 6,5,1,2
Face 5,4,0,1
Color 160,160,160

[MeshBuilder] ;sátortető
Vertex -2,3,4
Vertex 0,5,4
Vertex 2,3,4
Vertex 2,3,-4
Vertex 0,5,-4
Vertex -2,3,-4
Face 0,1,4,5
Face 4,1,2,3
Face 5,4,3
Face 2,1,0
Color 128,0,0
```



4. feladat – Létra a puszta közepén

```
; zöld fű
[MeshBuilder]
Vertex -20,0,20
Vertex 20,0,20
Vertex 20,0,-20
Vertex -20,0,-20
Face 0,1,2,3
Color 10,150,10

; égbolt
[MeshBuilder]
Vertex -20,20,-20
Vertex -20,20,20
Vertex -20,0,20
Vertex -20,0,-20
Vertex 20,20,20
Vertex 20,0,20
Face 0,1,2,3
Face 1,4,5,2
Color 120,120,220

;létra egyik szára
[MeshBuilder]
Cube 0.04,0.7,0.04
Color 80,45,0
Translate 0,0.7,-0.25

;létra másik szára
[MeshBuilder]
Cube 0.04,0.7,0.04
Color 80,45,0
Translate 0,0.7,0.25

;fokok
[MeshBuilder]
Cylinder 8,0.03,0.03,0.5
Color 80,45,0
Rotate 1,0,0,90
Translate 0,0.3,0

[MeshBuilder]
Cylinder 8,0.03,0.03,0.5
Color 80,45,0
Rotate 1,0,0,90
Translate 0,0.7,0
```

```
[MeshBuilder]
Cylinder 8,0.03,0.03,0.5
Color 80,45,0
Rotate 1,0,0,90
Translate 0,1.1,0
```