3D ábrázolás – PoVRay (3. rész)

Az előző részekben már használtunk néhány tömör testet, főként gömböt, amely mind közül az egyik legegyszerűbb. A gömbhöz hasonló egyszerű testeket primitíveknek nevezzük, ugyanis ezek az összetett testek kiinduló alkatrészei.

A *PoVRay* a testeket másképp készíti és használja, mint az elterjedt poligon alapú 3D programok. Ezen utóbbiak esetén a testek kisebb-nagyobb háromszögekből állnak, így természetüknél fogva üregesek és közelről nézve szögletesek. A *PoVRay* matematikai módszerekkel írja le a testeket, amelyek így tömör és sima tárgyakként viselkednek. Ennek főleg a textúrák alkalmazásakor lesz jelentősége. A tömör tárgyak nagyon sok esetben jól jönnek, de vannak esetek, amikor érdemes üreges testeket alkalmazni: a *PoVRay* azonban erre is képes.

Sík

A való világ tárgyai általában véges méretűek, vannak azonban olyan problémák, amelyeket végtelen kiterjedésű tárgyakkal leszünk képesek megoldani. Ilyen probléma például az égbolt, a sík föld vagy a tenger ábrázolása. Erre a `plane` alkalmas, amely egy végtelen síkot prezentál (*1. ábra, pov27.pov*):

```
plane
{
 y, 0
 texture{
  pigment{
   color White}}}
```

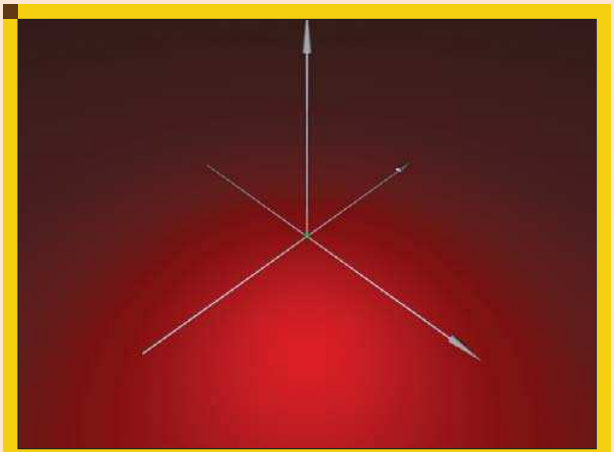
A sík egyik fontos paramétere a sík ***normálvektora***, amely a síkra merőleges és az origóból indul. Könnyedén tudjuk helyettesíteni a *PoVRay* belső neveivel, ugyanis az y név a `<0,1,0>` vektort fedi, s az ***x***, illetve a ***z*** is ezzel azonos módon képezhető. Ha ***y*** normálvektort adunk meg, akkor a létrejövő sík az ***x*** és a ***z*** tengely által meghatározott helyen jön létre.

```
plane
{
 y, -1
 texture{
  pigment{
   color White}}}
```

A síkot el tudjuk tolni a normálvektora mentén (*2. ábra, pov28.pov*), ehhez csupán a normálvektor után megadott számot kell a szükséges értékre módosítani, amely az origótól való távolságot határozza meg. A normálvektorral és az origótól való távolsággal tetszőleges síkot, illetve síkok kombinációját el tudunk készíteni. A *3. ábrán* (*pov29.pov*) három olyan síkot készítettünk el, amelyek egymással 60 fokos szöget zárnak be.

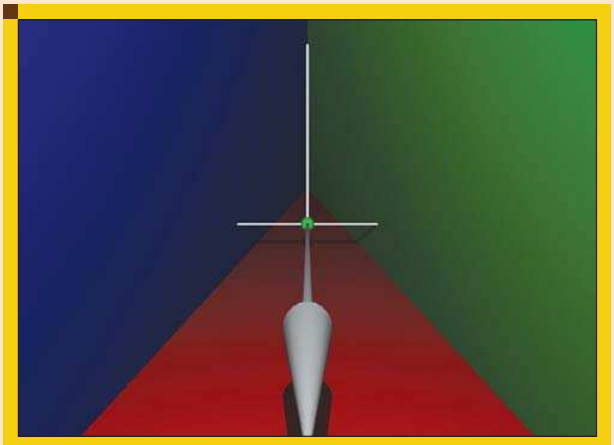
Doboz

Gyakori test a téglalap, vagy más néven a doboz, amelyet a két átellenes sarokpontjával tudunk meghatározni (*4. ábra, pov30.pov*).

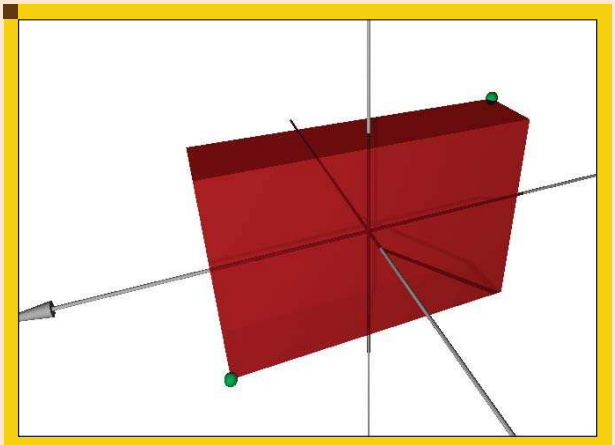


1. ábra A sík elhelyezkedése a koordináta rendszerben

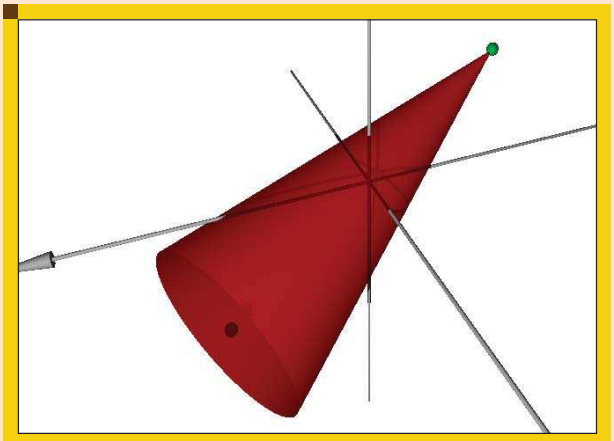
2. ábra A sík pozíciója



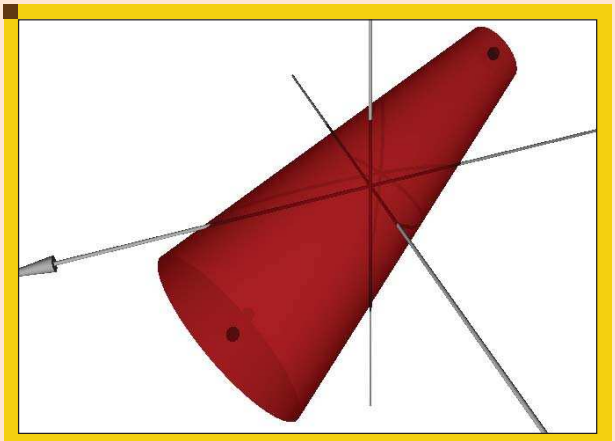
3. ábra Három sík, háromszögben



4. ábra Áttetsző téglatest



5. ábra Áttetsző kúp



6. ábra Áttetsző csonka kúp

Az ábrán egy vörös színű, félig átlátszó téglalapot láthatunk, amely a két meghatározó sarokpontjában egy-egy zöld gömböt tartalmaz.

```
box{
 <-5,3,-1>, <5,-3,1>
 texture{
  pigment{
   color <1,0,0,0.5>}}}
```

Bárhova is tesszük a sarokpontokat, a téglalap élei mindig párhuzamosak valamelyik tengellyel, így külön „trükköznünk” kell a megfelelő pozícióba forgatáshoz (de erről majd egy későbbi részben lesz szó).

Kúp és csonkakúp

Egy kúpot három paraméterrel tudunk meghatározni: a bázispontjával, a tetőpontjával és az alapjának sugarával (*5. ábra, pov31.pov*).

```
cone{
 <5,-3,1> 3, <-5,3,-1> 0
 texture{
  pigment{
   color <1,0,0,0.5>}}}
```

Ha csonka kúpot akarunk előállítani, akkor a tetőpontjánál nullától különböző sugarat kell írnunk (*6. ábra, pov32.pov*).

```
cone{
 <5,-3,1> 3, <-5,3,-1> 1
 texture{
  pigment{
   color <1,0,0,0.5>}}}
```

A kúp lehet üreges is, ha az `open` kulcsszót a megfelelő helyre írjuk, s ekkor a két lezáró kör alakú lemez nem kerül ábrázolásra (*7. ábra, pov33.pov*).

```
cone{
 <5,-3,1> 3, <-5,3,-1> 1
 open
 texture{
  pigment{
   color <1,0,0,0.5>}}}
```

Henger

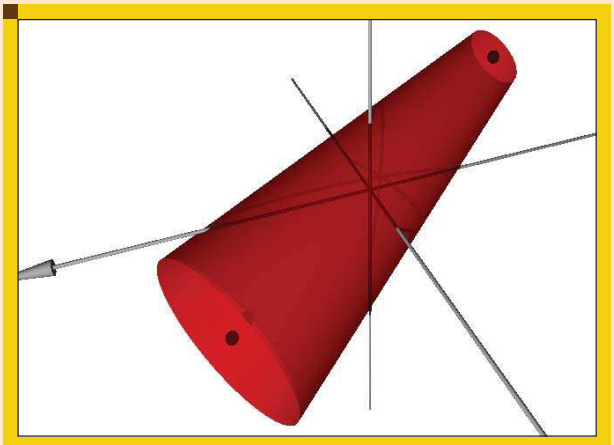
A henger teljesen azonosan kezelhető, mint a kúp, mivel a PoVRay szempontjából egy speciális csonka kúp (*8. ábra, pov34.pov*).

```
cylinder{
 <5,-3,1>, <-5,3,-1>, 2
 texture{
  pigment{
   color <1,0,0,0.5>}}}
```

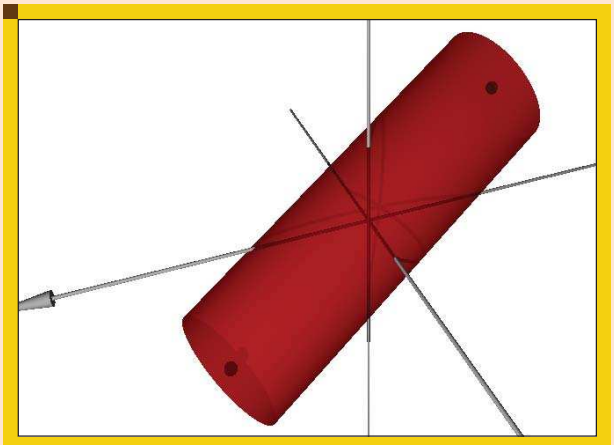
A henger lezáró korongjait az *open* kulcsszóval tudjuk eltávolítani.

Gömb

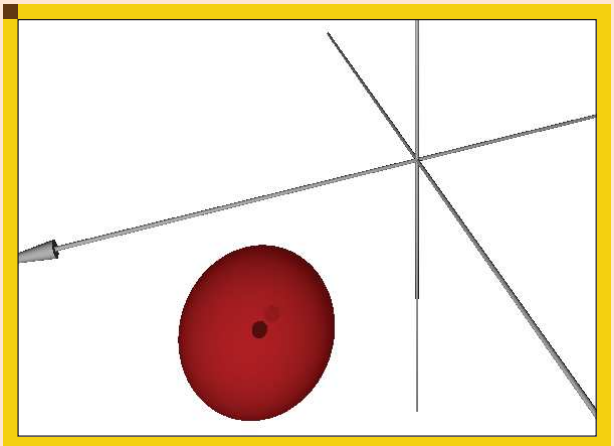
A gömb leírása a legegyszerűbb az összes test közül, a középpontját és a sugarát kell megadnunk (*9. ábra, pov35.pov*).



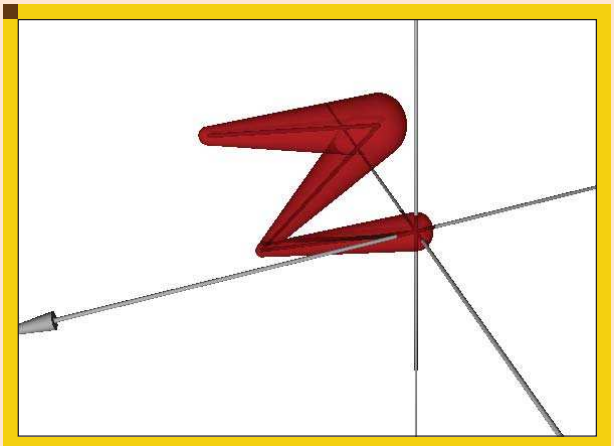
7. ábra Áttetsző üreges csonka kúp



8. ábra Áttetsző henger



9. ábra Áttetsző gömb



10. ábra „Suhanó" gömb, linear_spline útvonallal


```
sphere{
 <5,-3,1>, 2
 texture{
  pigment{
   color <1,0,0,0.5>}}}
```

„Suhanó" gömb

A gömb „egyenes ági" leszármazottja egy olyan test, amelyet a ***PovRay*** a gömb útvonalából és kontúrjából épít fel. Így tudunk olyan rudat készíteni, amelynek a két vége nem egy sima körlemez, hanem félgömb.
Ha több sarokpontot is megadunk, akkor a kontúr híven követi ezt az útvonalat (*10. ábra, pov36.pov*).

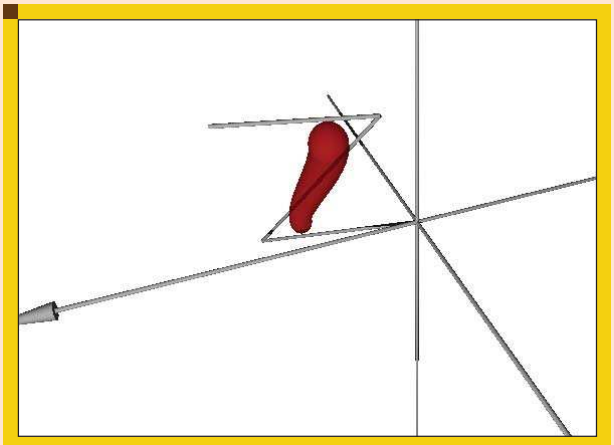
```
sphere_sweep{
 linear_spline
 4,
 <0,0,0>, 0.5,
 <4,-2,-4>, 0.2,
 <1,3,0>, 0.7,
 <5,3,-1>, 0.2
 texture{
  pigment{
   color <1,0,0,0.5>}}}
```

Számomra érthetetlen okból előre meg kell adnunk a sarokpontok számát, amely alapján a program megfelelő számú térbeli pontot és sugarat vár. Ezek előtt meg kell adni a sarokpontok közötti útvonalat leíró „egyenletet" is, amely igazából három megadható kulcsszóra egyszerűsödik számunkra (a konkrét megvalósítás a programozók dolga volt):

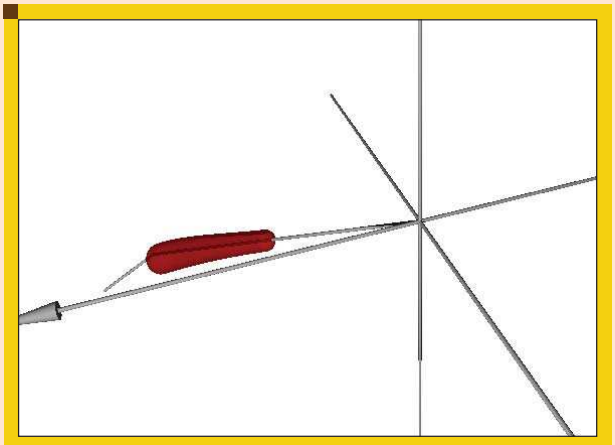
`linear_spline`, `b_spline`, illetve `cubic_spline`.

Az első útvonalra már láttunk példát, ekkor a sarokpontokat egy térbeli egyenes mentén köti össze a program és a sarokpontokba képzelt gömbök sugara egyenletes átmenettel jelenik meg.

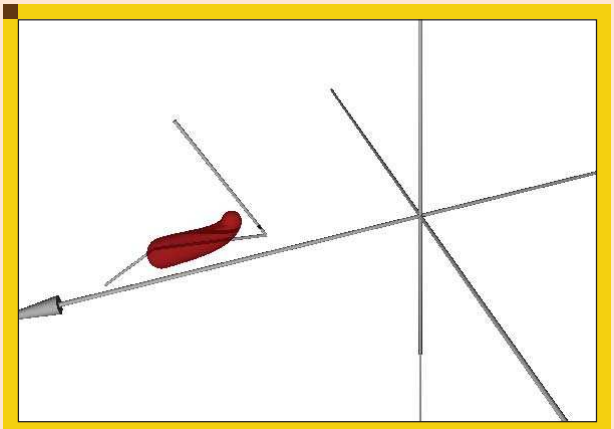
Ha a `b_spline` útvonalat választjuk, akkor a „suhanás" közel sem az egyenesek által meghatározott pontokon át történik. A létrejövő test a sarokpontokat nem érinti, hanem „tehetetlenül" mozog a sarokpontok irányába.
A *11. ábrán* (*pov37.pov*) látható testet kapjuk abban az esetben, ha a sarokpontok között a törések túl éles szögben történnek, így a „suhanás" túl nagy tehetetlensége okán nem tudja elérni a sarokpontokat.
Kicsit laposabb szögek esetén az eredmény sokkal látványosabb (*12. ábra, pov38.pov*). A szemfülesek észrevehetik, hogy az így létrejövő test sokkal rövidebb, mint az egyenes mentén mozgó gömb által létrehozott test. Ennek oka, hogy a `b_spline` (és a `cubic_spline`) az első és az utolsó sarokpontot a kezdő-lezáró „tekeredés" meghatározására használja. Ha a 12. ábrán látható útvonal első sarokpontját



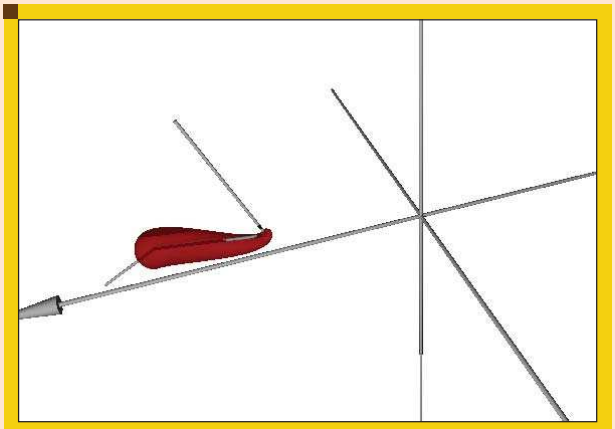
11. ábra „Suhanó” gömb, b_spline útvonallal és éles szögekkel



12. ábra „Suhanó” gömb, b_spline útvonallal és lapos szögekkel



13. ábra „Suhanó” gömb, b_spline útvonallal és éles kezdő szöggekkel



14. ábra „Suhanó” gömb, cubic_spline útvonallal és éles kezdő szöggekkel

merőlegesre állítjuk (*13. ábra, pov39.pov*), akkor látható, hogy a kezdőpont elmozdult az első sarokpont irányába.
A `cubic_spline` útvonal teljesen hasonló a `b_spline` által meghatározott útvonalhoz, csak az útvonal minden esetben érinti a közbenső sarokpontokat (*14. ábra, pov40.pov*).

„Cuppanós” testek

Kiválóan alkalmazható kémiai bemutatóhoz, illetve olyan esetekben, amikor gömbök vagy téglatestek lépnek egymással kölcsönhatásba. A „cuppanás” azt takarja, hogy ha a leírt testek közel kerülnek egymáshoz, akkor a felületük egyesül, mintha folyadékcseppek lennének. Minél közelebb kerülnek, annál jobban látszik ez az egyesülés. Példaképpen nézzünk meg egy vízmolekulát, amely egy oxigén atomból és két hidrogén atomból áll (*15. ábra, pov41.pov*).

```
blob{
threshold 0.6
sphere{<0,0,0>,4,1
texture{pigment{color Blue}}}
sphere{<1.15,-1.63,0>,1,1
➥texture{pigment{color Red}}}
sphere{<1.15,1.63,0>,1,1
➥texture{pigment{color Red}}}
scale 2}
```

Tórusz

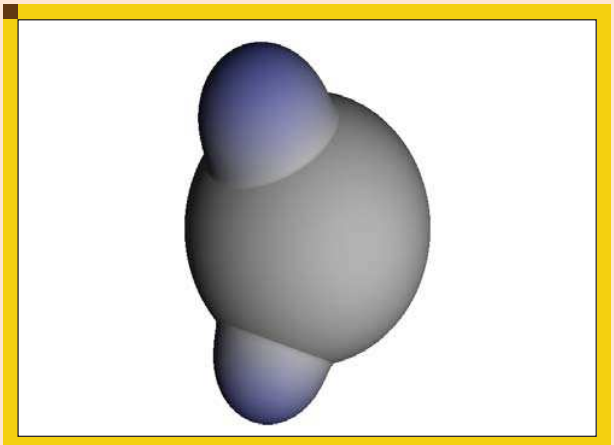
Ha egy gömböt végigvonszolunk egy kör alakú útvonalon, akkor egy tóruszt kapunk eredményül. Ezt megtehetjük a „suhanó” gömb segítségével is, de jobban járunk ha a PoVRay saját tóruszát használjuk (*16. ábra, pov42.pov*).

```
torus{
 7, 1.5
 texture{
  pigment{
   color <1,0,0,0.5>}}}
```

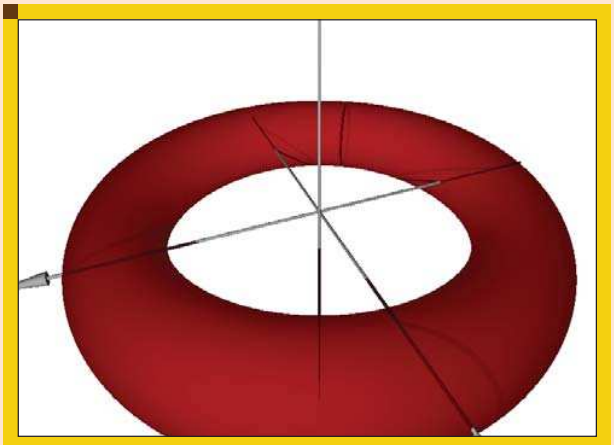
A tórusz mindig az origó által meghatározott középponttal jön létre, és két paraméter határozza meg: a fősugár (***major radius***) és a melléksugár (***minor radius***). A fősugár a kör alakú útvonal origótól való távolságát határozza meg, a melléksugár pedig a körbevonszolt gömb sugarával egyezik meg. Ha máshova és másképp szeretnénk tóruszt létrehozni, akkor koordináta transzformációval át kell helyeznünk, illetve át kell alakítanunk.

Magasságmező

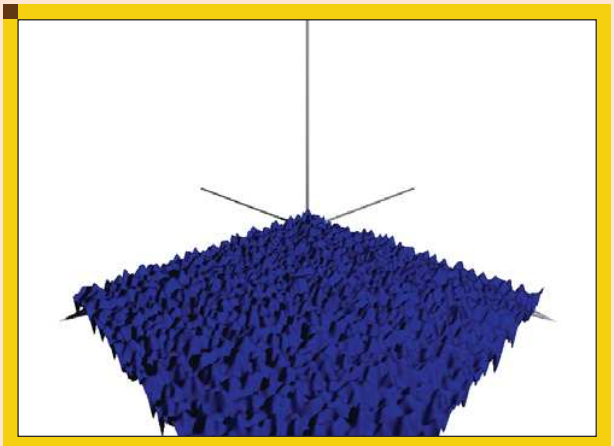
Ha hegyes-völgyes képet szeretnénk elkészíteni, azt nehezen tudjuk matematikai egyenletekkel leírni (bár nem lehetséges), így a ***PoVRay*** is csak háromszögekből képes ezt előállítani. Sok ezer háromszöget viszont nehéz megfelelő pozícióba helyezni, a magasság-mező használatával ez felesleges munka is lenne.



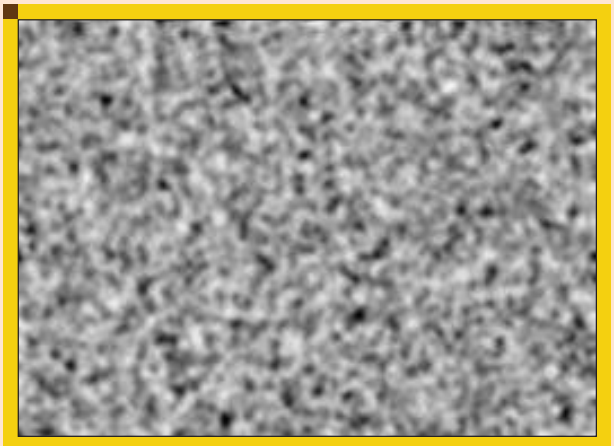
15. ábra „Cuppanó" gömbök, vízmolekula



16. ábra Tórusz



17. ábra Magasság mező



18. ábra A magasság mező forrása

A magasság-mező használatához kerítenünk kell egy olyan állományt, amely szürkeárnyalatos képet tartalmaz. A fekete területek a nulla szintet, a fehér területek az egységnyi magas szintet jelentik. A kettő közötti árnyalatok a világosságukkal arányos magasságot jelképeznek. Alapesetben a `<0,0,0>` és `<1,1,1>` pont által közbezárt téglalapon belül jön létre ez a test, ezért megfelelő transzformációkkal el kell tolnunk a kívánt pozícióba.

```
height_field{
 png
 "pov43field.png"
 smooth
 scale 10*x
 scale 10*z
 texture{
  pigment{
   color Blue}}}
```

Érdekes trükköket tudunk elérni, ha egy fényképet alkalmazunk a mezőhöz (*19-20. ábra, pov44.pov*).

Forgatott felszín (Surface of Revolution)

Gyakran találkozunk olyan testekkel, amelyeket egy meghatározott sík felület valamely tengely körül megforgatott kontúrja határoz meg. Tipikus és szép példa erre a serleg, amely ilyen forgatás eredménye (*21. ábra, pov45.pov*).

```
sor{
 11,
 <0,0>
 <7,0>
 <6,2>
 <2,3>
 <2,6>
 <3,7>
 <2,8>
 <2,14>
 <6,16>
 <7,21>
 <5,21>
 texture{
  pigment{
   color Blue}}}
```

A megforgatás mindig az ***Y*** tengely körül történik, és speciális ***U-V*** koordinátákkal kell megadni a pontokat. Gyakorlatilag az ***U*** az ***X*** irány, a ***V*** az ***Y*** irányt jelöli. Megadhatunk éles vonalakat is, mert a létrejövő felület simított és lekerekített lesz, így nem tűnnek fel durva élek a képen. Természetesen ezáltal éles körvonalakat nehezebb lesz létrehozni.

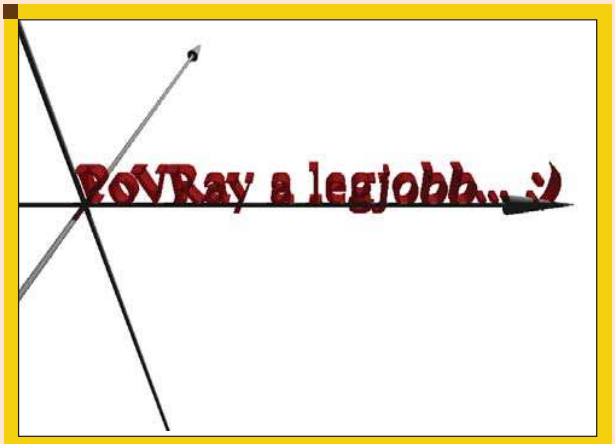
Szöveg kiírása

Néha szükséges szövegeket írni a ***3D*** világunkba, ezt is megtehetjük, készíthetünk betű kinézetű testeket, amelyek az írás szerint követik egymást (*22. ábra, pov46.pov*).

19. ábra Magasság mező

20. ábra A magasság mező forrása

21. ábra Megforgatott sík



22. ábra Szöveg kiírása

```
text{
 ttf "LucidaBrightRegular.ttf"
"PoVRay a legjobb... :)",
➥0.3, 0
 texture{
  pigment{
   color <1,0,0,0.5>}}}
```

A szövegnek négy paramétere van, amelyek rendre a betűtípusfájl elérési útja (ezt csak TTF lehet), a kiírandó szöveg, a szöveg térbeli kiterjedése, illetve a betűk közötti hely beállításához egy kétdimenziós vektor (ez utóbbi általában nulla). Ha ritkítani szeretnénk a betűket, akkor a `0.1*x` eltolás egész jó érték; ha sűríteni, akkor a `-0.1*x` a jó választás. Tudunk ferdén is írni, ha például a `<0,0.1,0.1>` eltolást használjuk, ekkor a karakterek *x* és *z* irányban `0.1` egységnyivel mozognak. (*23. ábra, pov47.pov*).

További testek

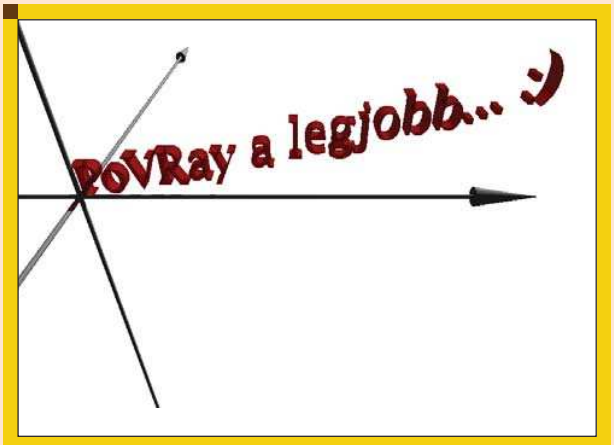
A ***PoVRay*** a felsorolt testeken kívül még egyszer ennyi lehetőséget rejt, azonban ezek használata kissé nehézkes és fárasztó lehet (fraktálok, különféle végtelen kiterjedésű testek, poligonok és alakzatok). Érdemes fellapozni a dokumentációt ezen testek megismerése végett, de csak akkor ajánlatos, ha a már ismert testekkel nem tudjuk ábrázolni az elképzelt világot.

Testek közötti halmazműveletek

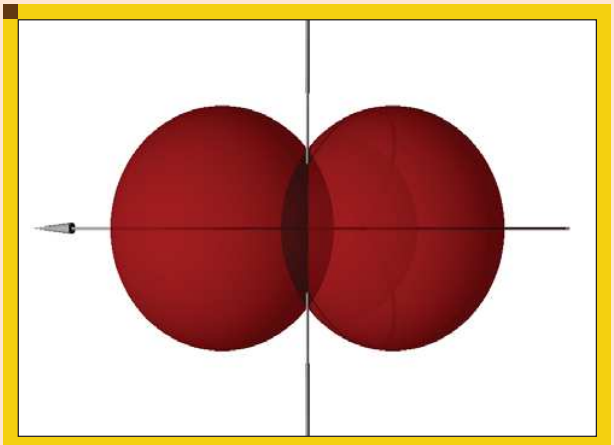
Önmagukban ezek a testek semmire sem jók, ha nem tudunk közöttük bizonyos átalakító műveleteket végezni. Ha ezekből a „primitív” testekből építkezünk, akkor szinte minden valós tárgyat el tudunk készíteni, amire szükségünk van.
Alapvetően a metszetet, a különbséget illetve a egyesítést tudjuk használni, azonban az egyesítésből két különböző módszert ajánl a ***PoVRay***. Az egyik egyesítés inkább technikai, ugyanis a testeknek minden része megmarad, az is, ami egymásba lóg (*24. ábra, pov48.pov*).

```
union{
 sphere{
  <-3,0,0>,4
  texture{
   pigment{
    color <1,0,0,0.5>}}}
 sphere{
  <3,0,0>,4
  texture{
   pigment{
    color <1,0,0,0.5>}}}}
```

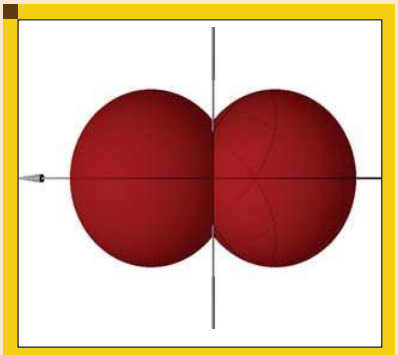
A másik típusú egyesítés a kontúrok mentén egyesíti a testeket, így nem marad egymásba lógó részük, viszont ez erőforrásigényesebb. Akkor célsze-



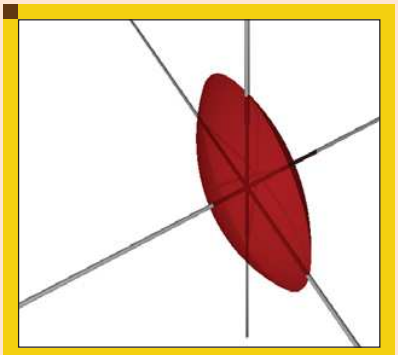
23. ábra Szöveg kiírása kis offszettel



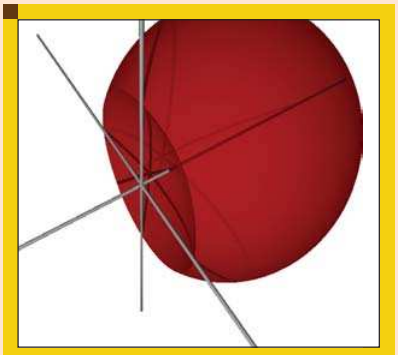
24. ábra Két gömb egyesítése, union



25. ábra Két gömb egyesítése, merge



26. ábra Két gömb metszete



27. ábra Két gömb különbsége

rű használni, ha átlátszó vagy áttetsző testeket használnunk (*25. ábra, pov49.pov*).

```
merge{
 sphere{
  <-3,0,0>,4
  texture{
   pigment{
    color <1,0,0,0.5>}}}
 sphere{
  <3,0,0>,4
  texture{
   pigment{
    color <1,0,0,0.5>}}}}
```

A *PoVRay* képes kettő vagy több test közös részét kiemelni, ezt nevezzük metszetnek. Ha a kettő gömb metszetét nézzük, akkor egy lencse alakú tárgyat kapunk (amelyet később a fénytöréssel és az átlátszósággal valódi lencseként is tudunk használni) (*26. ábra, pov50.pov*).

```
intersection{
 sphere{
  <-3,0,0>,4
  texture{
   pigment{
    color <1,0,0,0.5>}}}
 sphere{
  <3,0,0>,4
  texture{
   pigment{
    color <1,0,0,0.5>}}}}
```

Az utolsó művelet szerint egy testből kivonjuk azt a részt, amely egy másik testtel közös: ezt nevezzük a két test különbségének (*27. ábra, pov51.pov*).

```
difference{
 sphere{
  <-3,0,0>,4
  texture{
   pigment{
    color <1,0,0,0.5>}}}
 sphere{
  <3,0,0>,4
  texture{
   pigment{
    color <1,0,0,0.5>}}}}
```

A következő részben a testek különféle transzformációival fogunk tüzetesebben foglalkozni, így kényelmesen össze tudjuk állítani az elképzelt világot primitív testekből.

Auth Gábor
(auth.gabor@enaplo.hu)

Egy pécsi középiskolában informatikát és programozást oktat. Tíz éve botlott először a UNIX rendszerekbe, 7 év Linux használat után kapta el a FreeBSD lázat, amiből máig nem tudott kigyógyulni.

KAPCSOLÓDÓ CÍMEK

A PovRay projekt honlapja
➲ http://www.povray.org

A cikkben említett fájlok
➲ http://user.enaplo.hu/~auth.gabor/pov/