PoV-Ray ismeretek (8. rész)

Sorozatunk utolsó részében a függvények ismeretéből fakadóan már az animációk elkészítésének rejtelmeibe pillanthatunk bele.

Tekintettel arra, hogy ezt a programot igazából a sugárkövetéses képszámítás használatára készítették, nem találkozhatunk benne olyan megoldásokkal, melyeket az általánosan használt modellezőkben megszokhattunk. Itt nincs mód hierarchikus tárgyfelépítésre, ha tehát valaki például inverz kinematikát szeretne használni, és nem riad vissza a nehézségektől sem, még akkor is sokáig eltart, amíg az animáció elkészültekor minden tárgy a helyére kerül. Ennyi bevezetés után lássunk végre munkához!
Egyszerű animációt fogunk elkészíteni, amelynek egy képét a sorozat előző részében már felvázoltam. Most a malom lapátkerekét, a vizet és a felhőket fogjuk mozgásra bírni, miután példákat mutatok az előzőekben alkalmazott dinamikus tárgylétrehozásra. A jelenet a következő tárgyakra bontható:

- a malom épülete,
- mellette a malomkő,
- a táj,
- az égbolt,
- a víz – felette pára,
- a kerítés.

Természetesen a malom épülete nem csupán egy objektumból áll, hanem egy nagyobb `box` objektumból kivontam egy kisebbet, majd ebből az ajtó és az ablakok helyén szintén kivonással készítettem el a nyílásokat. Az ablakkeretek külön tárgyak, ezeket végül az épület alsó részével egyesítettem. A tetőszerkezetet háromszögalapú `prism` objektumokból képeztem, majd a homlokzatot egy külön `prism` objektum alkotja. Itt el lehetett volna hagyni a homlokzatot alkotó harmadik tárgyat, de a szemléltetés kedvéért megtartottam. Így a tető létrehozása után például rácsos homlokzatot is készíthetünk, amely jobban szellőzik.
Most részletesebben kitérek az ajtó elkészítésének módjára, bár nem olyan nagy ördöngösség. Az ajtóléceket egy ciklusutasítás (`#while`) segítségével egy keskeny téglatest eltolásával hozom létre. A ciklusszámláló változásakor a számlálót használom a pillanatnyi eltolás kiszámításához. Itt minden ajtólécet a léc szélességével és egy tetszőleges értékkel kell eltolni, amely az ajtólécek közötti rést határozza meg. Az ajtóléceket egymás mellé helyező kódrészletet az *1. listán* láthatjuk.
Hasonló módon készült a kerítés is, ez esetben viszont a közbeeső vastagabb oszlopok létrehozására a maradékos osztás műveletét is felhasználtam. Amikor a ciklusváltozó maradék nélkül osztható öttel, az oszlopot nem kell annyira lekicsinyíteni, így a kerítésben oszlopok és keskenyebb tartólécek is lesznek. A kerítés két részből áll, mindkettő ezzel a módszerrel készült, de a ház előtt lévő rész rövidebb, kevesebb tartóoszlopból áll, és a létrehozás után még el kellett fordítani, hogy a másik lécsorra merőlegesen álljon.
A malom lapátkerekét szintén több részből készítettem, CSG-műveletek felhasználásával. A lapátokat tartó abroncs két henger különbségeként alakult ki. A kerékagyat és a küllőket

1. lista

```
// itt jon az ajto
#declare ajtolec = 0
#declare Ajto=union {
  #while ( ajtolec < 10 )
    box {
      <0, 0.07, 0>
      <0.07, 1.48, -0.05>
      translate <-0.08*ajtolec, 0, 0>
    }
    #declare ajtolec = ajtolec + 1
  #end
}
```

2. lista

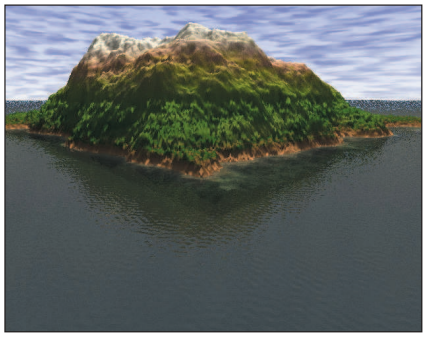
```
// Max. ennyire lehet nyitva az ajt
#declare max_ajto=40

// Ennyire van kinyilva az ajto
#if (clock<0.5)
  #declare ajto_nyilasszoge =
           max_ajto*clock
#else
  #declare ajto_nyilasszoge =
           max_ajto-(max_ajto*clock)
#end

// Az ajt  Øs forgatÆsa
object {
  Ajto
  material {
    AjtoMat
  }
  rotate -y*ajto_nyilasszoge
  translate -0.16*x
  translate -0.95*z
}
```

egyszerű hengerek alkotják, a küllőket oldalirányból egy kicsit összenyomtam. E helyütt a lapátokat és a küllőket szintén ciklusban hoztam létre. A lapátoknál még ki kellett számítani az egyes lapátok új helyzetét és az elfordulásukat. Ezeket a koordinátákat szinusz és koszinusz függvényekkel lehet kiszámítani. Az x koordinátát a következő képlet alapján: x=R*cos(rad(szög))+dx, míg az y koordinátát az y=R*sin(rad(szög))+dy képlet segítségével. Az `R` annak a körnek a sugara, amelynek mentén a tárgyat el szeretnénk

helyezni, míg a dx és a dy értékek e kör a középpontját határozzák meg. A képletekben szereplő sz g természetesen az elforgatás szöge fokban meghatározva. Az egyes darabok elkészítése után a részeket egyesítettem, így a további eltolások és forgatások már az egész kerékre vonatkoznak.
A malomkő esetében a legnehezebb feladat a pontos elhelyezés és dőlés beállítása volt – maga a tárgy két hengerből készült, szintén kivonással. Az elhelyezését kísérleti úton oldottam meg, hiszen így tűnt a legegyszerűbbnek.



A hangyák szigete

Techno-malom

A táj létrehozásakor a jól ismert Gimp nevű képfeldolgozó és festőprogramot hívtam segítségül, és két képet készítettem vele. Az egyik egy szürkeárnyalatos kép, melyet először középszürke (R: 128; G: 128; B:128) színűre festettem be, utána pedig sötétebb színnel a mélyebb területeket (például a patakmeder és a tó (ez az animációban nem látszik)) festettem rá, majd világosabb színekkel kialakítottam a dombokat és hegyeket. A táj színeinek és mintázatának meghatározásához később ezt a képet használtam fel. Első lépésként 256 színűvé alakítottam egy zöldárnyalatokból álló paletta használatával, majd visszaalakítottam 24-bites színmélységűvé. Így már rendelkezésemre állt egy zöld, sárga és szürkeárnyalatokból álló tájkép felülnézete. Erre külön rétegekben ráfestettem a patakokat, majd az egyéb árnyaló színeket, hogy a kép ne legyen annyira egyszínű. A harmadik rétegen festettem meg az utat, ennek elkészítése vette igénybe a legtöbb időt. Minden próbálkozás után kiszámítottam egy képet, és ha a ház nem volt teljesen az út leágazó részének a végén, az utat arrébb kellett festenem, mert a patak miatt az épületet nem lehetett elmozdítani a helyéről. Végül a részletek jobb láthatóságáért a képet a kétszeresére nagyítottam és újabb változatos mintákat festettem a fűre.
Az előbbi tájkészítési mód bonyolultnak tűnhet, ezért el kell mondanom, hogy csak azért volt szükség a kézi munkára, hogy a tárgyakat pontosan el tudjam helyezni. Mivel nagyméretű tárgyakról van szó, amelyeket nem lehet pontatlanul elhelyezni, teljesen sík felületet kellett létrehoznom, ahol a síkalapú épületet el tudtam helyezni. A CD-mellékleten található néhány segédprogram, köztük a terraform, amely kifejezetten felülnézeti domborzati képek készítésére használható. Korábbi DOS-os és más operációs rendszer alatt futó megfelelője VistaPro névre hallgat, bár a CD-n található program tudásában kissé elmarad a nem linuxos változattól.
A program a létrehozott domborzatból PoV-Ray forrást is képes készíteni, amit a későbbiek folyamán saját céljainkra használhatunk fel. A fenti képen ezzel a programmal készítettem egy tájat, amit szintén a PoV-Ray számított ki.
A CD-mellékleten (28. CD Magazin/Pov-Ray) találhatunk még egy átalakító programot, amellyel különféle 3D-formátumokat tudunk konvertálni, többek között *.pov* forrásállománnyá, valamint egy kifejezetten a PoV-Rayhez készülő modellező (TrueVision) programot is, mely ugyan még fejlesztésének kezdeti szakaszában van, de bonyolultabb jeleneteink megalapozásához így is alkalmazható. A TrueVisionnal készítettem el a példában szereplő jelenet anyagait, majd a beállításokat a hagyományos módon (szövegszerkesztővel) finomítottam.
Mindezek után rátérhetünk a jelenet mozgó részeinek meghatározására. A PoV-Rayben egy animáció elkészítésekor a program a bemeneti forrásállományt újra és újra beolvassa és feldolgozza, majd a memóriában létrehozza a tárgyakat, az anyagokat, a fényforrásokat stb. Ez azt jelenti, hogyha valamilyen változást szeretnénk előállítani, akkor a két képkocka előállítása közben kell a forrásban módosításokat végezni. Például megtehetnénk ezt úgy is, hogy egy külső program minden képkockához előállít egy forrásállományt, amelyek csak néhány sorban és vektorösszetevőben különböznek egymástól, majd a PoV-Ray segítségével kiszámítjuk őket. Ez nem túl kényelmes megoldás, ezért a programozók egy belső változót bocsátanak a rendelkezésünkre, a clock-ot, melynek értékét a PoV minden képkocka kiszámítása előtt növelni fogja. Hogy mennyivel fog változni képről képre, a következők alapján számíthatjuk ki: a PoV-Ray számára megadhatunk egy kezdőképszámot, egy másik értékben pedig a képek legnagyobb számát. A képkockák kezdeti indexét a Initial_Frame, az utolsó kocka számát a Final_Frame értékkel határozzuk meg a beállításokat tartalmazó fájlban. Ugyanígy határozzuk meg az időszámláló kezdő- és végértékét a Initial_Clock és Final_Clock kulcsszavakkal. A clock minden képkocka között a következő képlettel meghatározható értékkel fog változni:

```
Delta_clock=(Final_Clock-Initial_Clock)/
➥(Final_Frame-Initial_Frame)
```

Ezt a változást használjuk fel a jelenetleíró állományban, hiszen mindig azonos értékkel fog módosulni, és a segítségével tetszőleges tárgyat is nagyon egyszerűen elforgathatunk. Erre az elforgatásra mutat egyszerű példát az alábbi lista:

```
cylinder {
    0*y,
    3*y,
    0.2
    rotate z*clock*360
}
```

Itt jegyezném meg, hogy a clock-változó kezdő- és végértékét célszerű 0-ra és 1-re állítani, mert így könnyebben számolhatunk vele. Erre a beállításra is láthatunk példát a CD-mellékleten (28. CD Magazin/Pov-Ray), de az érthetőség kedvéért itt is megemlítem:

```
Input_file_name=malom.pov
Initial_Frame=0
```

```
Final_Frame=120
Initial_Clock=0
Final_Clock=1
```

A fentiek szemléltetésére lássunk egy bonyolultabb példát is (a 2. *lista* tartalmazza), melyet a cikk fő témáját adó jelenetből vettem.
Itt a feltételes utasítás-végrehajtást a mozgást szolgáló `clock`-változóval együtt alkalmaztam, így az animáció első felében az ajtó negatív irányba fordul el, a második felében pedig az ellenkező irányba, vagyis kinyílik és becsukódik. Az `Ajto`-t mint tárgyat egymás mellé helyezett lécek egyesítésével már korábban meghatároztam.
Ezekkel az ismeretekkel bárki nyugodtan elindulhat a PoV-Rayjel készített animációk világának felfedezésére. Az alábbiakban egy képet mutatok be, ehhez hasonló látható az előző oldalakon is. A különbség csak annyi, hogy ez esetben már nem egy ember görbe vonalai alkotják a ház körvonalait, hanem gépies pontosságú egyenesek. A számítógéppel előállított kép számos esetben kisebb hatást gyakorol az emberi lélekre, mint egy hagyományos módon festett kép, hiszen a festményeknél nem csupán a tintapettyek halmaza hat ránk, hanem egy bizonyos szinten a festő érzéseit, gondolatait is átélhetjük. A számítógéppel előállított képek esetében ezt a közvetítő szerepet a megfelelően megalkotott jelenettel kell helyettesíteni, ezért úgy gondolom, számítógéppel nehezebb hatásos képet alkotni, mint ecsettel és festékkel. A CD-mellékleten (28. CD Magazin/Pov-Ray) a befejező részhez tartozó animáció MPEG-2 formátumban megtalálható, és egy segédprogram is, amivel az egyes képeket animációvá fűztem és tömörítettem. Mivel a PoV-Rayt bemutató sorozat a végéhez érkezett, már csak egy hibajavítást szeretnék végrehajtani.
A sorozat hatodik részében a `halo` anyagtípusról adtam leírást, de a PoV-Ray újabb változataiban ez a típus már nem szerepel, használata hibát okoz, és a számolás már az előfeldolgozás során leáll. Helyette a `media` kulcsszó alkalmazható, ami több beállítási lehetőséggel rendelkezik, és gyakorlatilag a programhoz adott példákon keresztül, kísérletezéssel lehet leginkább megismerni. Az írásunkhoz készült példában is alkalmazom a patak fölött lévő pára megvalósításához.
Ennek kiindulási alapja a CD-mellékletre is felkerült ***hollow2.pov*** forrásállomány volt. Annak ellenére, hogy a PoV-Ray telepítőcsomagja ezeket a példákat tartalmazza, mégis felkerültek a CD-re, mert így könnyebben megtalálhatók és tanulmányozhatók.
Köszönöm az érdeklődést, és további kellemes perceket kívánok a PoV-Rayjel végzett munkához!

Fábián Zoltán
(dzooli@freemail.hu, dzooli@yahoo.com)
24 éves, jelenleg programozóként dolgozik. Szabadidejében szívesen kirándul, túrázik. Emellett szeret rajzolni, érdekli a 3D grafika és a Linuxszal kapcsolatban minden olyan program és programnyelv, amit még nem ismer vagy nem próbált ki.