© Kiskapu Kft. Minden jog fenntartva

PoV-Ray ismeretek (1. rész)

Cikksorozatunkban a PoV-Ray sugárkövető programot mutatjuk be.

Ebben a részben áttekintjük a PoV-Ray főbb szolgáltatásait, megismerkedünk a legkisebb gépkövetelményekkel, elkészítjük az első egyszerű jelenetünket, majd összefoglaljuk a kamera és a fényforrások meghatározásának lehetőségeit.

Talán azzal illene kezdenem, hogy mi is a PoV-Ray. Szabadon használható program, mely igen sok szolgáltatással rendelkezik. Fotószerű képeket készíthetünk a sugárkövetés eljárásának felhasználásával. A PoV-Ray a *David K. Buck* és *Aaron A. Collins* által készített DKBTrace 2.12-es változatán alapul, melyet a szerzők szabadidejükben alkottak meg.

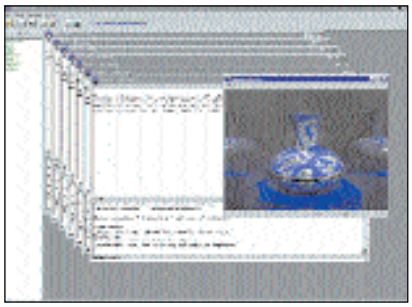
A programmal létrehozott jeleneteket szövegszerkesztővel fogjuk elkészíteni, a jelenetleíró nyelv írásmódja hasonló a C nyelvben megszokottéhoz. Természetesen nem kell mindent kézzel beírnunk, mert számos előre meghatározott alakzattal dolgozhatunk, és amit egyszer már megalkottunk, azt a későbbiekben felhasználási helyén egyszerűen beilleszthetjük. Alakzatok mellett színállandókat és előre meghatározott mintázatokat is tartalmaz a PoV-Ray-csomag, továbbá a program képes kétdimenziós képek alapján domborzat megjelenítésére is, ehhez látványos köd-, szivárvány- és atmoszférahatásokat alkalmazhatunk. A modellek szerkezetét CSG (Constructive Solid Geometry)-módszerrel is meghatározhatjuk, ami nagymértékben megkönnyíti a bonyolultabb modellek elkészítését. Ebben az esetben az egyszerű alakzatokból (gömb, kocka, tórusz stb.) logikai műveletekkel állítjuk elő a számunkra megfelelő formát. Ha például egy domború lencsét szeretnénk meghatározni, akkor ezt úgy is megtehetjük, hogy gömböt metszünk el egy kockával, majd az eredményül kapott testet lemásoljuk, és a másolatot szembefordítjuk az eredetivel. Megfelelő elhelyezés után már készen is van a domború lencse. A program lehetőséget ad számos egyedi, különleges hatás létrehozására is, például: tűz, felhőzet, füst, valamint poros levegőben láthatóvá váló fénysugár megjelenítése. Mintázatként a sokfajta beépített mintázat (például: márvány, leopárd mintázat, bump mapping, színátmenet, pepita mintázat) mellett használhatunk képeket is.

A kimeneti fájlformátumok között a következők találhatók meg: PNG, TGA és BMP, valamint Machintosh felületen .PIC formátumban is tárolhatjuk a képeket. IBM-PC felületen a kiszámolt képeket 15- és 24-bites színmélységben jeleníthetjük meg. Az elkészült képeket legfeljebb 48-bites színmélységben tárolhatjuk.

Mindezek mellett a program számos felületen elérhető, köztük Linuxon, Amigán, MS-DOS-on, Sun operációs rendszeren, Windowson és Apple Machintoshon.

➲ www.provray.org

➲ www.xlcus.co.uk/povray/island/pan.html



➲ www.xlcus.co.uk/povray/

Erőforrásigénye – a leírás szerint – a mai normákhoz képest igen szerény, a legnagyobb igényeket Windows alatt futtatva tapasztalhatjuk, de itt is megelégedhetünk 16 MB memóriával és egy 486 DX-es processzorral. Linuxon megelégszik 386-os processzorral, 4 MB memóriával és 10 megányi szabad tárhellyel. A gyorsabb futtatáshoz ajánlott Pentium processzor és 32 MB memória, de természetesen minél nagyobb teljesítményű a gépünk, annál hamarabb várhatunk eredményt.

Linux alatt a PoV-Ray minden hivatalos változatnak részét képezi, ha azonban valakinek mégsem lenne meg, akkor a ➲ http://www.povray.org címen elérheti, a Download csoportban letöltheti. Ennyi bevezető után lássunk egy a programmal készített képet, utána még közelebbről megismerkedünk a lehetőségekkel.

Első képünk elkészítéséhez tekintsünk át néhány alapfogalmat, amelyek megértése a további munkához és a példákhoz is elengedhetetlen. Legfontosabb, hogy a PoV-Ray koordinátarendszerét értsük meg. Ez „balkezes” koordinátarendszer, az Y tengely mutat felfelé. Leginkább úgy szemléltethetnénk, ha bal kezünk mutatóujjával mutatnánk felfelé és a középső-, hüvelykujjunk segítségével derékszögű koordinátarendszert képeznénk. Ilyen helyzetben a Z tengelyt jelképező gyűrűsujjunk a nézőponttól távolodó irányba mutat.

A jelenetek elkészítéséhez a Nedit szövegszerkesztőt használtam kitűnő makrózási lehetőségei következtében, felhasználva a lemezmellékleten található előre elkészített, dot.nedit fájlban található makrókat is. Ezt az állományt felhasználói könyvtárunkba kell bemásolnunk .nedit néven.

A következő lista alapján megérthetjük a PoV-Ray jelenetleíró nyelvének alapvető írásmódját és a jelenet felépítését.

1. lista A PoV-Ray alapvető írásmódja és a jelenet felépítése

```
#include "colors.inc"

global_settings
{
        ambient_light 0.7
}

// --------------------------- Kamera
camera
{
  location  <0.0, 0.5, -4.0>
  look_at   <0.0, 0.7,  0.0>
}

// --------------------------- Fenyforras

light_source {
        <4,4,-4>
        1.2*White
}

// ---------------------------- Alaplap
plane {
        y, -1
        pigment {
                checker pigment {Black},
                     pigment {White}
                scale <2,2,2>
                }
}

// ---------------------------- Hatso fal
plane {
        -z, -20

        pigment {
                image_map { gif "gekko.gif" }
                scale 15.9
                translate y*-1.5
                }

}


// --------------------------- Egy gomb
sphere
{
  0.0, 1
  texture {
          pigment {
             radial
             color_map {
              [0.0 rgbt <0, 0.3, 0, 0.2>]
              [0.5 rgbt <0.2, 0.2, 0.3, 0.2>]
              [1.0 rgbt <0.3,0,0,0.2>]
             }
             frequency 8
             turbulence 0.9
          }
          finish {
                  specular 0.6
                  ambient 0.6
                 }
          }
  rotate x*90
}
```

© Kiskapu Kft. Minden jog fenntartva

Amint azt a lista alapján is láthatjuk, a jelenet tartalmaz egy kamerát, ami meghatározza a megjelenítendő kép értékeit (oldalarányait), a nézőpontunkat és a kamerával használt objektív értékeit. Minden jelenet magába foglal legalább egy fényforrást, bizonyos esetekben ez megegyezik a környezeti szórt fénnyel – ez nem kézzelfogható fényforrás, de megvilágítja a jelenetet. Amennyiben egy jelenetben semmilyen fényforrást nem határozunk meg, akkor a PoV-Ray alapértelmezett környezeti megvilágítással számítja ki a képet.

Most pedig lássuk a kamera meghatározását és fontosabb értékeinek értelmezését! Az írásmódot a PoV-Ray leírásában található formában adom meg. Vektort a programban <x, y, z> alakban kell megadni, a lebegőpontos és egész értékek megadása pedig azonos a más programnyelvekben megszokottal. A programba beépítve találhatjuk meg az egységvektorokat mindhárom irányban, tehát ha valamilyen helyet vagy irányt nem akarunk teljes koordinátákkal megadni, akkor ezeket az egységvektorokat használhatjuk a megfelelő szorzótényezővel. Például: a <5.2,0,0> vektor egyenértékű a következővel: x*5.2.

Elsőként vegyük sorra a használható objektívtípusokat:

Perspective

Ez a klasszikus objektív, amit a fényképezőgépekben is alkalmazunk. A vízszintes látószöget egyrészt a „direction” vektor és a „right” vektor hosszúságának aránya határozza meg, másrészt az „angle” érték segítségével is meghatározható. Ez utóbbi használata egyszerűbb. A látószög értéke 0 és 180 fok között lehet.

Orthographic

Ebben az esetben párhuzamos vetítést használunk, a kép oldalarányait pedig az „right” és az „up” vektorok hosszúságának aránya szabja meg. Például a klasszikus 4:3 arányú megjelenítéshez „up y” és „right 1.33*x” vektorok megadásával juthatunk.

Fisheye

Halszemoptika, gömbszerű vetítéssel. Ez a típus is ismerős a fényképészet területéről, a látószöget az „angle” kulcsszóval adjuk meg. 180 fokos látószöggel a szokásos halszemoptikának megfelelő hatást érhetünk el, míg 360 foknál minden körülöttünk lévő tárgyat láthatunk. Ezen optikával kör vagy ellipszis alakú képet kapunk eredményül.

Ultra_wide_angle

Hasonlít a halszemoptikához azzal a különbséggel, hogy mindenképpen téglalap (vagy négyzet) alakú képhez jutunk. A látószöget szintén az „angle” kulcsszó segítségével határozhatjuk meg.

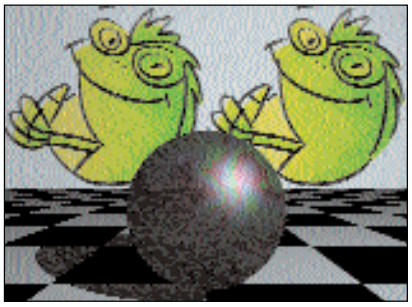
Omnimax

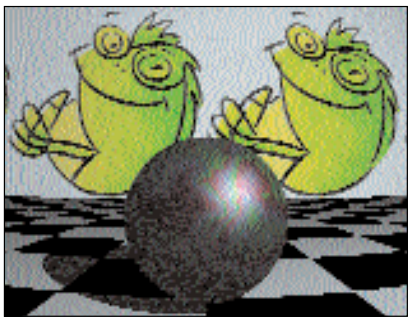
Ez a vetítési mód egy 180 fokos halszemoptikánál használtnak felel meg, de a vízszintes látószög kisebb és nem változtatható az „angle” kulcsszóval sem. Ilyen képeket (filmeket) használnak a kupola alakú vetítővászonnal rendelkező úgynevezett Omnimax filmszínházakban.

Panoramic

Panorámaképek előállításához használható, mert kiküszöböli azt a perspektív vetítésnél tapasztalható hibát, amit a 180 fokot megközelítő látószögeknél tapasztalunk. Itt szintén alkalmazható az „angle” érték.

© Kiskapu Kft. Minden jog fenntartva

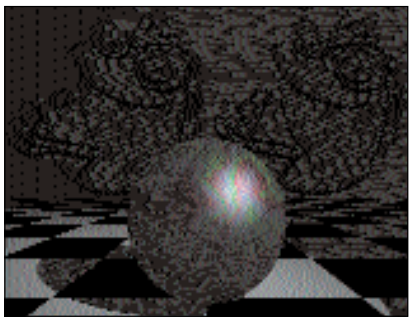


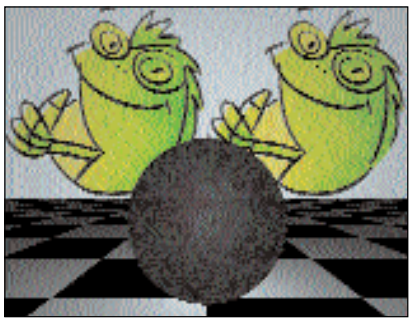


Cylinder

Ebben az esetben a jelenetet egy henger felületére képezzük le, a megadott valós érték 1, 2, 3 vagy 4 lehet. 1-es értéknél a hengerfelület függőleges állású, 2-es értéknél vízszintes, 3-as értékkel szintén függőleges irányú felületet adunk meg, de itt a nézőpont helyzete a henger tengelyén mozog, míg a 4-es érték mozgó nézőpontot és vízszintes tengelyű hengert határoz meg. A kamera látószögére vonatkozóan csak a perspektív vetítésnél van megkötés (0–180 fok közé kell esnie). Különleges hatásokat érhetünk el, ha 360 foknál nagyobb látószöget adunk meg, ennek hatására a képen a jelenet többször fog megjelenni. Hogy pontosan hányszor láthatunk ismétlődést, az a kamera egyéb értékeitől is függ, így érdekes lehetőség nyílik az ismerkedésre, próbálgatásra.

A kamera helyzetét a „location” kulcsszóval adhatjuk meg, míg a célpont helyét, ahová majd kameránk néz, a „look_at” kulcsszóval. Mindkettő egy-egy vektort vár értékként. Az „up” és „right” vektorok szerepe többnyire az, hogy meghatározzák a kamera irányát, de ezt sokkal könnyebben megtehetjük a „direction” és a „look_at” értékek használatával. Ennek ellenére ez a két vektor mégis fontos, mert ezekkel adjuk meg a kép oldalarányait. Az egyes vetítési típusoknál más-más értelmezéssel kell számolnunk. Perspektív, panoráma és nagy látószögű (ultra_wide_angle) objektív használatakor a vektorok meghatározzák a kép arányait és a „direction” vektorral együtt a látószöget is. A „direction” helyett ajánlott az „angle” használata. Párhuzamos vetítés esetén szintén e két vektor arányának megfelelő képet kapunk, de itt a „direction” értékét a program nem veszi figyelembe, mint ahogyan a halszemoptikánál sem. Hengerfelületre való vetítést alkalmazva, annak 1-es és 3-as típusánál a henger tengelyének iránya megegyezik az „up” vektor által kijelölt iránnyal, és a 3-as típusnál ennek a vektornak a hosszúsága meghatározza azt is, hogy a tér mely koordinátái között kell a képet kialakítani. Például „up 4*y” azt jelenti, hogy csak az y=-2 és y=2





közötti pontok lesznek láthatók. Ezeknél a típusoknál minden vetítősugár merőleges lesz az y tengelyre. A 2-es és 4-es hengeres vetítési típusnál a hengerfelület tengelye párhuzamos a „right” vektorral és minden vetítősugár merőleges az x tengelyre.

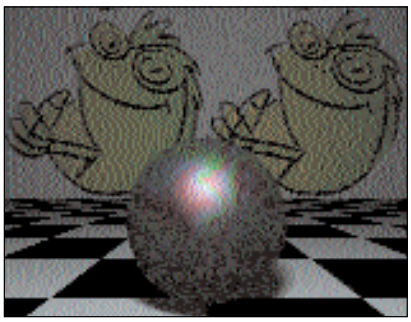
Nem szabad elfelejtenünk, hogy a „right”, az „up” és a „direction” vektoroknak egymásra merőlegesnek kell lenniük, hogy elkerüljük a torzulásokat. Ha ez a helyzet nem áll fenn, a számítás elindításakor egy figyelmeztető üzenetet kapunk.

Az „angle” értéket, mint az a korábbiakból is kitűnik, a látószög meghatározásához használjuk, így most csak a pontosítás kedvéért adom meg az összefüggést a „right”, a „direction” vektorok hosszúsága és a látószög között:

```
látószög= 2 * arctan(0.5 *
        ➥right_hosszúsága /
        ➥direction_hosszúsága)
```

A PoV-Ray képes mélységélesség szimulálására is, ami azt jelenti, hogy a kamera fókuszában lévő terület élesebb lesz, mint a fókuszon kívül eső részek. Ennek pontosabb meghatározására alkalmas az „aperture” (rekesz) kulcsszó: nagyobb értékek esetén az élesebb terület kisebb lesz, míg kisebb rekeszt használva az élesebb terület mérete megnő. Az éles terület középpontját a „focal_point” (fókuszpont) vektor segítségével adjuk meg. Ezt célszerű a jelenet fő témájára állítani. Alapértelmezésben a PoV-Ray nem számol mélységélességgel, tehát ezt nekünk kell bekapcsolni az „aperture” és a „blur_samples” értékek megfelelő megválasztásával. A „blur_samples” értéke határozza meg azt, hogy az életlen területek elmosásához a program hány környező fénysugár átlagát vegye figyelembe.

Miután megismerkedtünk a kamera meghatározásával és használatával, tekintsük át a négy legfontosabb fényforrás általános formáját és a különböző értékek használatát. A fényforrás helyét egy vektorral adjuk meg, a színét pedig egy „rgb” kulcsszó után szereplő három 0-1-ig terjedő valós értékkel (piros, zöld és kék összetevők mértéke).



Például: color <rgb 0.3, 0.3, 0.3>

A „shadowless” kulcsszó alkalmazásával az adott fényforrás nem képez árnyékokat. Az „adaptive” kulcsszó után szereplő egész érték a mintatúlvételezést befolyásolja, ennek eredményeképpen az éles árnyékok elmosódnak. Az 1-es értéknél 4, a 2-es értéknél 9, a 3-as értéknél 25, míg 4-es értékű mintatúlvételezéssel a program 81 környező pont átlagaként fogja meghatározni egy-egy pont színezetét.

Mivel a fényforrásoknak alapértelmezésben nincsen megjelenési formájuk, szükség lehet arra, hogy a fényforrást megfeleltessük valamilyen tárgynak, vagyis „formába

2. lista A kamera általános alakja a fontosabb értékekkel

```
camera {
    [ perspective | orthographic |
      fisheye | ultra_wide_angle |
      omnimax | panoramic |
      cylinder FLOAT ]
    location <VEKTOR>
    look_at <VEKTOR>
    right <VEKTOR>
    up <VEKTOR>
    direction <VEKTOR>
    right <VEKTOR>
    angle VALÓS
    blur_samples VALÓS
    aperture VALÓS
    focal_point <VEKTOR>
  }
```

3. lista A pontszerű fényforrás általános formája

```
  light_source {
    <F NYFORRAS_HELYE>
    color <SZ"N>
    [ shadowless ]
    [ adaptive EGESZ ]
    [ looks_like { OBJEKTUM } ]
    [ atmospheric_attenuation BOOL ]
  }
```

4. lista A fényszóró fényét egy pontra irányítjuk

```
light_source {
       <0,0,0>
       color < rgb 1, 0.5, 0.5 >
       spotlight
       radius 10
       falloff 20
       fade_distance 8
       fade_power 1
       point_at <1,2,1>
}
```

5. lista Jól látható a téglalap alakú területet megvilágító fényforrás használata

```
light_source {
        <0,4,-4>
        1.2*White
       area_light x,y,4,4
       fade_distance 5
       fade_power 1
}
```

öntsük". Ilyen eset lehet például egy gyertya lángjának a megjelenítése. A „looks_like" kulcsszó pontosan ezt a célt szolgálja, a megadott objektum a fényforrás alakjaként fog megjelenni.
Az objektumot előre kell meghatározni és a „no_shadow" kulcsszó használatával ki kell zárni az árnyékot vető tárgyak közül azért, hogy az objektum árnyéka ne takarja el a jelenet többi részét. Erre láthatunk példát a lemezmellékleten található tutor1_5.pov fájlban.
A PoV-Ray alapértelmezésben nem számol azzal a ténnyel, hogy a fény szétszóródik a ködben és az atmoszférán áthaladva veszít erősségéből. Az „atmospheric_attenuation" bekapcsolt állapotában (atmospheric_attenuation on) azonban ezt a szolgáltatását is használhatjuk, látványos hatásokat érve el vele.
A következő fontos fényforrás a spotlámpa. Ez a valóságban a fényszórónak felel meg. A fentieken felül további öt értéke lehet. Meghatározásához a szín beállítása után meg kell adnunk a „spotlight" kulcsszót, majd a fényszóró által állandó erősséggel megvilágított szögtartományt a „radius" kulcsszó után. Lehetőségként használható az is, hogy az egyenletes megvilágításon túl még hány fokos tartományban világítson a lámpa. Ezt a „falloff" kulcsszóval és a hozzá tartozó valós értékkel határozhatjuk meg. A „fade_distance" az egyenletes megvilágítás fényforrástól való távolságára van hatással, vagyis minél nagyobb ez az érték, annál messzebbre világíthatunk az erősség csökkenése nélkül. Az erősség csökkenésének módját adja meg a „fade_power" érték. Ha ez 1, akkor a fény erőssége lineárisan csökken, 2-es értéknél pedig négyzetesen. A fényszóró fényét pontosan egy pontra irányíthatjuk a „point_at" kulcsszó után megadott vektor segítségével.
Ugyanilyen értékkel használhatunk henger alakú fénycsóvát is a megvilágításra, ekkor a „spotlight" kulcsszó helyett a „cylinder"-t kell alkalmaznunk. Míg a fényszóró kúp alakú térrészt világít meg, a hengeres fényforrás fénye henger alakú térrészben érzékelhető.
Az első rész utolsó témájaként egy olyan fényforrást ismertetek, amely egyaránt használható nagyobb területek, valamint a környezeti megvilágítás megvalósítására.
A környezeti szórt fény kiindulási pontja nem mindig határozható meg pontosan, előfordulhat, hogy pontosan ilyen fényforrásra van szükségünk. Itt tehát az „area_light"-ról lesz szó, ennek alapvető értékei megegyeznek a spotlámpánál leírtakkal, kivéve a „radius" és a „falloff" kulcsszavakat, melyek itt nem értelmezhetők. Mivel ez a típus téglalap alakú területet világít meg, kiegészítésképpen megadható a téglalap oldalainak aránya és a fényforrás két tengelye.
Ha jobban megfigyeljük az egyes fényforrások hatására kialakuló árnyékokat, akkor észrevesszük, hogy ezzel a típussal érhetünk el igazán valósághű árnyékokat, itt ugyanis már nem jelentkezik a pontszerű fénynél tapasztalható éles szélű árnyék.
Ezzel elérkeztünk sorozatunk első részének végéhez, remélem, hogy ezzel a rövid ízelítővel sikerült felkelteni az érdeklődést e nagyszerű program iránt. A további fejezetekben tárgyaljuk az alapvető testeket, az anyagok és mintázatok alkalmazását, az összetettebb alakzatok modellezését, az animáció lehetőségeit, végül néhány hasznos segédeszközt is bemutatok, amelyekkel megkönnyíthetjük munkánkat.
A lemezmellékleten található példák segítik a fentiek megértését, kiszámoltatásukhoz a következő parancssori értékekkel indítottam a PoV-Ray-t:
#povray -i*példafájl_neve*.pov +w*Szélesség* +h*Magasság* +L/usr/lib/povray3/include
Az utolsó érték (+L*könyvtárnev*) azt az elérési utat adja a program tudtára, ahol a beillesztendő fájlok találhatók. Erre az útvonalra a colors.inc miatt van szükség.
Várom az érdeklődők kérdéseit a következő levélcímre ➲ dzooli@freemail.hu.
A listák megtalálhatók a 14. CD-melléklet Magazin/Cikkekhez könyvtárában.

Fábián Zoltán
(dzooli@freemail.hu, dzooli@yahoo.com)
23 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.

© Kiskapu Kft. Minden jog fenntartva