A Blender és a Yafray tuningolása

A legtöbb esetben nincs szükségünk arra, hogy forráskódból telepítsünk egy programot, és nincs is mindig értelme. De mikor lehet szükségünk rá mégis? Ha nagy számításigényű programot futtatunk és szeretnénk a legjobban optimalizált kódot használni, hogy a gépünk lehetőségeit a legjobban kihasználjuk. Vagy ha szeretnénk az adott programból a legfrissebb verziót használni, esetleg nem akarjuk megvárni míg megjelenik a disztribúcióhoz tartózó csomag. A Blender esetében mind a kettő erősen szerepet játszik, ezért személy szerint kizárólak forrásból használom.

A forráskód és a szükséges programok beszerzése

A *Blender* forráskódját sokféleképpen szerezhetjük be. Közvetlenül letölthetjük a *CVS*-ből, vagy a stabil verziót csomagolt formában. Ez vonatkozik a *Yafray*-ra is. Mi az a *CVS*? A program fejlesztésére használt verzió kezelő, változás követő rendszer. A programozok ezen keresztül fejlesztik a programot. Ha ezt használjuk, akkor biztosak lehetünk benne, hogy a program legfrissebb verzióját használjuk. A csapda ott van, hogy a program éppen fejlesztés alatt van, ezért rengetek hiba lehet a programban, sőt az is előfordulhat, hogy nem tudjuk lefordítani. Természetesen nem kell választanunk a kettő között, kipróbálhatjuk mindegyiket akár szimultán is. Érdemes ellátogatni a következő címre ➲ http://www.blender3d.org/cms/The_Release_Cycle.361.0.html ahol részletesen le van írva, hogyan működik a *Blender* fejlesztése.
A mindenkori stabil forrást pedig erről a helyről tudjuk letölteni ➲ http://download.blender.org/source/.
Hozzunk létre egy külön könyvtárat a manipulációnkhoz és tömörítsük ki ebbe a mappába a forráskódokat:

```
$ mkdir blender_cvs_src
$ cd blender_cvs_src
$ tar xzvf blender-2.41.tar.gz
```

vagy *CVS*-t használva:

```
$ cvs -d:pserver:
➥anonymous@cvs.blender.org:/cv
➥sroot/bf-blender login
$ cvs -z3 -d:pserver:
➥anonymous@cvs.blender.org:/cv
➥sroot/bf-blender co blender
```

vagy egy nem hivatalos *CVS* ág esetén (több újdonságot tartalmaz, gyorsabban frissül):

```
$ cvs -d:pserver:
➥anonymous@cvs.blender.org:/cv
➥sroot/tuhopuu login
$ cvs -z3 -d:pserver:
➥anonymous@cvs.blender.org:/cv
➥sroot/tuhopuu co tuhopuu3
```

A *Yafray* esetén hasonló módon:

```
$ cvs -d:pserver:
➥anonymous@cvs.blender.org:/cv
➥sroot/yafray login
$ cvs -z3 -d:pserver:
➥anonymous@cvs.blender.org:/cv
➥sroot/yafray co yafray
```

Illetve a *Yafray* stabil forráskódját töltsük le a ➲ http://www.yafray.org/sec/2/downloads/ címről.
Ha ezzel megvagyunk akkor birtokunkban van a két forráskód. Persze még kellenek az egyéb függőségek is, amiket az *1. táblázat* tartalmaz.
Nagy valószínűséggel ezek már mind fenn vannak a gépünkön, de ha nem akkor telepítsük fel őket. Persze ha csomagból telepítjük őket, akkor tegyük fel a hozzá tartozó *dev* és *lib* csomagokat is ha vannak! Ezenkívül fontos hogy a videokártya meghajtója jól legyen feltelepítve.

1. táblázat *A fordításhoz szükséges függőségek*

Python 2.	http://www.python.org
libjpe	http://www.ijg.org
libpng	http://www.libpng.org/pub/png/
zlib	http://www.gzip.org/zlib/
Scons	http://www.scons.org
CVS	http://ximbiot.com/cvs/cvshome/
Ode	http://opende.sourceforge.net/
Openal	http://www.openal.org/home/
sdl	http://www.libsdl.org/index.php
smpeg	http://www.loki-games.com/development/smpeg.php3
fmod	http://www.fmod.org/
Glut	http://www.opengl.org
Freetype2	http://www.freetype.org/

Ez az *nVidia* esetén azt jelenti, hogy a *64 bites* telepítés során ne telepítsük fel a 32 *bites* kompatibilitási fájlokat, mert zavart okozhatnak. Ha nem szeretnénk a fenti függőségekkel egyenként bajlódni, akkor használjuk a következő parancsok valamelyikét.

Gentoo operációs rendszer esetén:

```
$ emerge -pv blender
```

vagy *Debian* operációs rendszer esetén:

```
$ apt-get install blender
```

Ekkor megtudhatjuk, hogy milyen programok kellenek a telepítéshez, és a *Blendert* illetve a *Yafrayt* kihagyva könnyedén fel tudjuk őket telepíteni, vagy *Blender-el* együtt telepítjük az egészet, majd külön *uninstalláljuk* a *Blendert* és a *Yafrayt*. Szóval most már van egy fordításra kész operációs rendszerünk.

Beállítás és fordítás

Lépjünk be a *Blender* forráskódot tartalmazó mappába, ami a mi példánknál a következő:

```
$ cd ./blender
```

Mi a *Scons* fordítási metódust fogjuk használni, mert ez a leginkább támogatott. A *Scons*-nek a *Sconstruct* nevű fájl mondja meg, hogy milyen paraméterek mellett kell elvégeznie a fordítást. Ebben a fájlban az operációs rendszernek megfelelően egy hivatkozást találunk, ami a további speciális beállításokat tartalmazza. *Linux* esetében ez a fájlt a *./config* mappában *linux2-config.py* néven található meg. Mindkét fájl nagyon beszédes, ezért könnyedén ki lehet igazodni és bátran fel lehet fedezni az egyéb beállításokat is, mert én csak a fordítóra vonatkozókat fogom tárgyalni. Tehát nyissuk meg a *linux2-config.py* fájlt a kedvenc szövegszerkesztőnkkel és az alábbi sorokat módosítsuk a következőkre:

```
111 CFLAGS = ['-w', '-pipe','
➥-fPIC','-funsigned-char', '
➥-ffast-math', '-fno-strict
➥-aliasing', '-fomit-frame-
➥pointer', '-finline-functions']
112 CCFLAGS = ['-w', '-pipe','
➥-fPIC','-funsigned-char', '
➥-ffast-math', '-fno-strict-
➥aliasing', '-fomit-frame-
➥pointer', '-finline-
➥functions']
114 CPPFLAGS = ['-DXP_UNIX']
115 CXXFLAGS = ['-w', '
➥-pipe','-fPIC','-funsigned-
➥char', '-ffast-math', '-fno-
➥strict-aliasing', '-fomit-
➥frame-pointer', '-finline-
➥functions']
116 REL_CFLAGS = ['-march=k8',
➥'-O3']
117 REL_CCFLAGS = ['-march=k8',
➥'-O3']
```

Ha 2.41 vagy ennél korábbi verzióval próbálkozunk, akkor nem lesz *linux2-config.py* fájlunk, sőt *./config* mappánk sem, ezért mindezen beállításokat a *SConstruct* fájlban kell elvégezni a következő módon:

```
79 release_flags = ['-
march=k8', '-O3']
80 debug_flags = ['-O2', '-g']
81 extra_flags = ['-w', '-
pipe','-fPIC','-funsigned-
char', '-ffast-math', '-fno-
strict-aliasing', '-fomit-
frame-pointer', '-finline-
functions']
```

Lássuk, hogy mi mit jelent?
Az *-march=* opcióval lehet megmondani a *gcc*-nek, hogy milyen processzorunk van, milyen mikro-utasításkészlettel fordítsa a programot (*mmx*, *sse*, *see2*, *3dnow*, stb.).
Az egyenlőségjel után kell írni, hogy milyen processzorunk van. Lássunk néhány példát:

AMD64 = k8
Pentium4 = pentium4
AthlonXP = athlon-xp

Ha valakit érdekelnek a további processzortípusokra vonatkozó opciók, akkor látogasson el a ➲ http://www.freehackers.org/gentoo/gccflags/flag_gcc3.html oldalra, ahol további részleteket tudhat meg. Az *-OX* opcióval pedig meg lehet mondani, hogy milyen optimalizációs szinten akarjuk a binárist létrehozni. Lehetséges értékei: *X=0, X=1, X=2, X=3 és X=s*. Az *'s'* érték azt jelenti, hogy nem gyorsaságra igyekszik a fordító, hanem minél kisebb méretre. Értelem szerűen mi a *3*-as értékre pályázunk. A többi opció mind speciális optimalizációt jelent, amire nem térek ki.
Amikor ilyen durva optimalizációs paraméterezéssel dolgozunk, nem kell meglepődnünk azon, ha nem fordul le a program, vagy ha le is fordul tele lesz hibával Ezért mindig le kell tesztelnünk a programot az alap beállításokkal is, hogy biztos nem hozunk-e létre hibás kódot, amiből a későbbiekben problémánk lenne. Ha *CVS*-t használunk, akkor ez fokozottan igaz, mivel itt állandó a fejlesztés, csak a stabil kiadás ideje alatt, vagy kódfagyasztáskor stabil a kód. Ezért érdemes mindenképpen egy optimalizáció nélküli binárist létrehoznunk, azt elmenteni valahova, esetleg gyorsan leteszteltni, hogy amikor hibák jönnek elő könnyen szét tudjuk választani a program hibát az optimalizációból fakadó hibától. Lássunk erre egy példát. Én a következő hibát találtam a *CVS* kód lefordításakor:
Mind a két fordítás ugyan azokkal az optimalizációs paraméterekkel készült, de mégis jól látható a különbség a kettő között. Az *1. ábrán* fekete foltok jelentek meg az üveg majmon, míg a *2. ábrán* szép kék, olyan amilyennek lennie kell. Ilyen és ehhez hasonló apró vagy nagyobb hibákat kell keresnünk a tesztelés során. A fenti *scene* egyébként a hivatalos tesztcsomagban a *render/* mappa alatt található *refract_monkey.blend* néven. Ha készen vagyunk a *linux2-config.py* fájlt szerkesztésével, akkor mentsük el és kezdjük meg a fordítást. Ehhez adjuk ki a következő parancsot a *Blender* főkönyvtárán belül:

```
$ scons
```

1. ábra Blender 2.41 CVS 2006 február

2. ábra Blender 2.41 tar.gz

2. táblázat *A kétféle tesztkörnyezet*

Teszt környezet	CPU	RAM	VGA	Operációs rendszer	Blender forrás
1	AMD64 3500+	1 GB OCZ	ASUS 6600GT	Gentoo 2005.1 64 bit	CVS
2	Celeron 700 MHz	128 SDRAM	Geforce 2 Noname	Ubuntu 5.10 i386	Office

3. táblázat *A teszteredmények*

Beállítások	Teszt környezet	Idő/s		
		-O2 -pipe	-O3 full opti.	Gyorsulás/%
OSA 8	1	51,74	49,52	4,30
640x480	2	151,85	127,91	15,77
OSA 16	1	130,85	125,04	4,40
1024x768	2	759,37	635,31	16,34
OSA 16	1	517,23	495,39	4,20
2048x1536	2	2983,60	2491,88	16,48

Addig amíg a *Blender* fordul, lépjünk be a *Yafray* főkönyvtárába, majd az itt talált *linux-settings.py* nevű konfigurációs fájlt módosítsuk a következőképpen:

```
$ cd ../yafray

11 prefix = args.get
➥('prefix','/usr')
19 flags='-Wall -DHAVE_CONFIG_H
➥-D_PTHREADS'
20 if debug:
21          flags+=' -O3
➥-ffast-math -ggdb'
22 else:
23          flags+=' -O3
➥-march=k8 -w -pipe -fPIC
➥-funsigned-char -ffast-math
➥-fno-strict-aliasing -fomit-
➥frame-pointer -finline-
➥functions'
24 return flags
```

És kezdjük meg a *Yafray* fordítását is:

```
$ scons
$ su
$ scons install
```

Ez szintén elfog tartani egy darabig, addig végezzük el az ügyes bajos dolgainkat. A kedves olvasó bizonyára észrevette, hogy a *Blender-nél* nem adtuk ki a scons install parancsot, csak a *Yafray-nál*. Ez nem véletlen, a *Yafray* esetében ha nem adjuk ki az `install` parancsot is, akkor nem tudjuk elérni a *Blender-ből*! Ha netalántán hibát kapnánk, vagy egyszerűen csak új fordítást szeretnénk csinálni más paraméterekkel, akkor az új fordítás előtt adjuk ki a következő parancsot, hogy tiszta lappal indulhassunk:

```
$ scons clean
```

Ez a parancs a *Yafray* esetében nem működik, helyette kézzel kell csinálni a dolgot:

```
$ rm ./src/Yafraycore/*.os
```

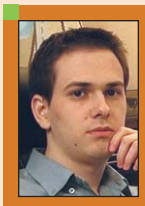
Ha szeretnénk a kész programot csomagolt formában megkapni, hogy könnyebb legyen a szállítása ill. telepítése, akkor adjuk ki a következő parancsot *Blender* esetén:

```
$ scons release
```

Ekkor a végeredményt nem a főkönyvtárban találjuk, hanem az összes szükséges fájlal együtt a *./dist* vagy újabban a *../install/linux2* könyvtáron belül. Így nem kell azon gondolkodnunk, hogy vajon milyen fájlok kellenek a *Blender* binárison kívül. Ha készen vagyunk a *Blender* fordítással, akkor tulajdonképpen futtathatjuk a programot bárhonnan, nem szükséges a $ scons install paranccsal telepíteni. Én csináltam egy *blender/* mappát a *home*/-on belül és ott futtatom. Ilyenkor kényelmesen futtathatunk akár több változatot is.

A tuningolás eredménye

Teszt fájl neve: *refract_monkey.blend*
A szokásos információkon kívül egyértelműen látszik a *3. táblázat* eredményeiből, hogy a sebességnövekedés a felbontás növelésével sem szűnik meg. Továbbá kisebb teljesítményű gépen a sebességnövekedés sokkal jelentősebb. Ez legrosszabb esetben is azt jelenti, hogy ha van egy 30 perces animációnk amit *PAL* szabványnak megfelelően 25 képkocka/másodperces és egy képkocka *renderelése* körülbelül 10 másodperc, akkor az animációnk körülbelül 5 órával előbb lesz kész. Talán a *3D CG*-re igaz a legjobban, hogy nincs az a teljesítmény, amivel az ember megelégedhetne. Remélem sikerült ezzel a módszerrel még egy kevés tartalékot kipréselni az otthoni gépekből.



Fábián Attila
(fabiana@elte.hu)
Az Eötvös Loránd Tudományegyetem Természettudományi karán vagyok vegyészhallgató. 4 éve használok Linuxot. Ha ezek után még mindig marad szabadidőm, akkor 3D grafikával foglalkozok vagy biciklizek.

KAPCSOLÓDÓ CÍMEK

A Blender webhelye:
➲ www.blender.org

A Yafray webhelye:
➲ www.yafray.org

Optimalizációs teszt:
➲ http://mail-index.netbsd.org/current-users/1995/12/11/0000.html

Optimalizációs paraméterek:
➲ http://www.delorie.com/gnu/docs/gcc/gcc_10.html

Tesztfájlok:
➲ http://download.blender.org/demo/test/test240.zip