A CodeX eddig már több mint 220 cikkel jelentkezett!

11.2004

CodeX

NEMZETKÖZI SZÁMÍTÁSTECHNIKAI - PROGRAMOZÓI SZAKLAP

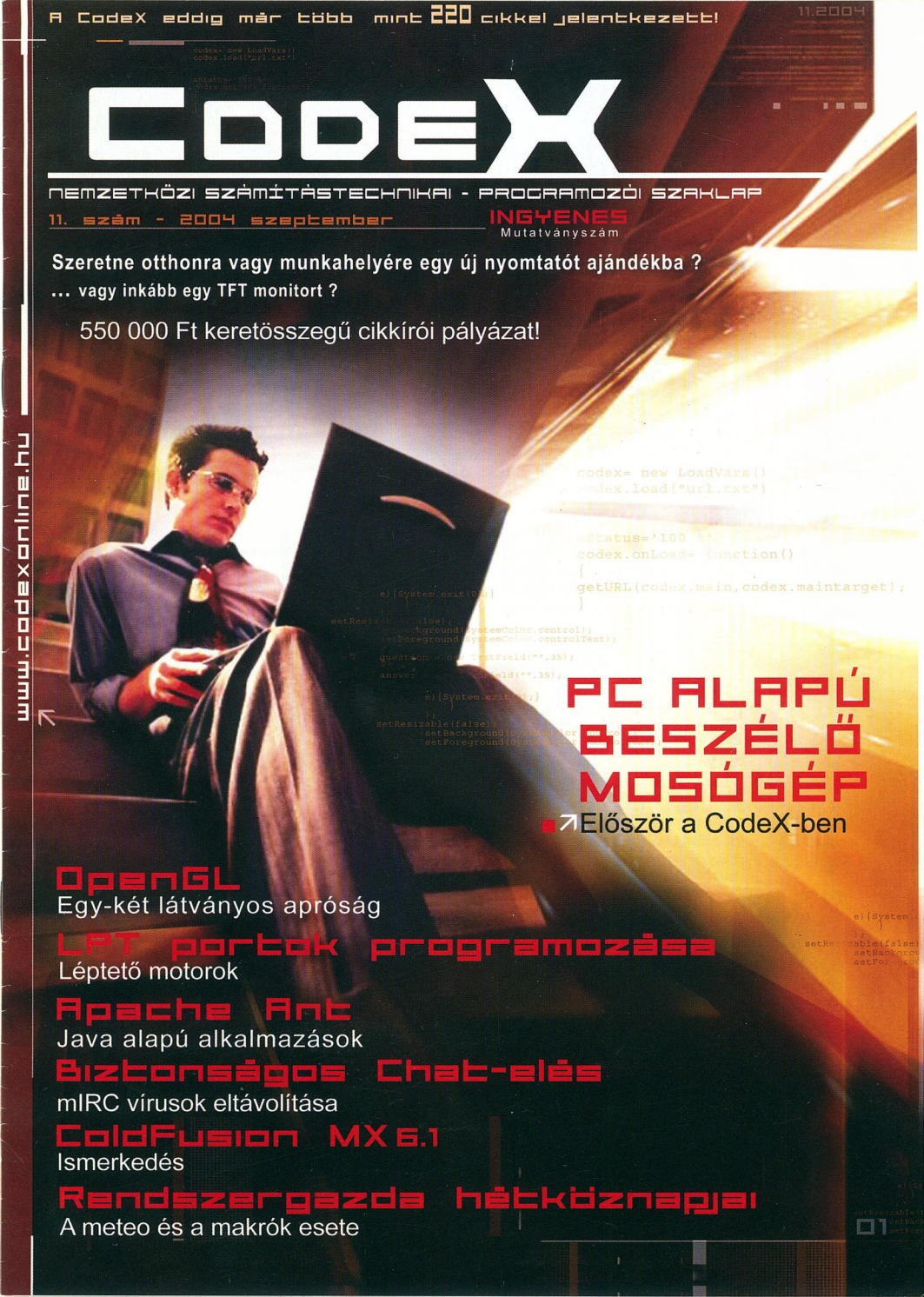
11. szám - 2004 szeptember

INGYENES
Mutatványszám

Szeretne otthonra vagy munkahelyére egy új nyomtatót ajándékba ?

... vagy inkább egy TFT monitort ?

550 000 Ft keretösszegű cikkírói pályázat!



PC ALAPÚ BESZÉLŐ MOSÓGÉP

Először a CodeX-ben

OpenGL

Egy-két látványos apróság

LAT portok programozása

Léptető motorok

Apache Ant

Java alapú alkalmazások

Biztonságos Chat-elés

mIRC vírusok eltávolítása

ColdFusion MX 6.1

Ismerkedés

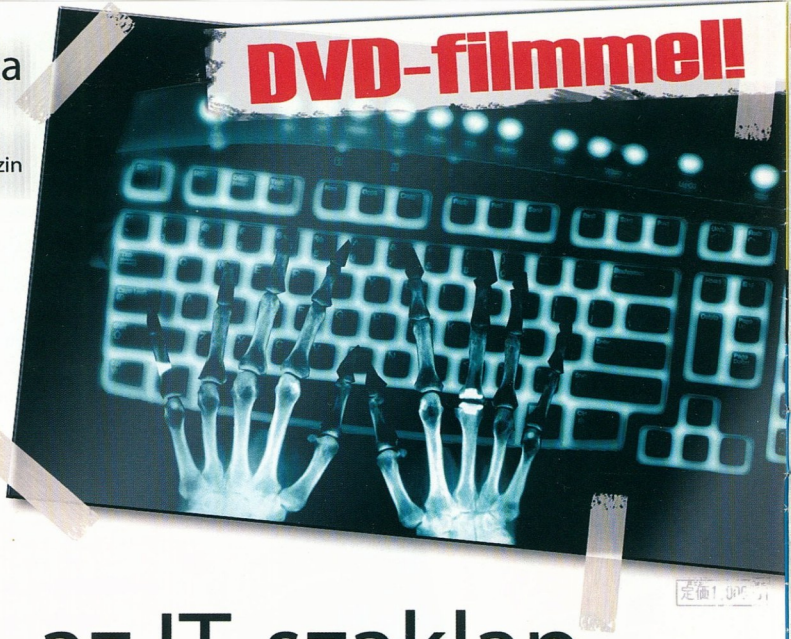
Rendszergazda hétköznapjai

A meteo és a makrók esete

www.codexonline.hu

„Az informatika átlátható."

– magyar pc magazin



PCM magyar ...az IT-szaklap

Testreszabott informatikai megoldások

INTERNETES ALKALMAZÁSOK

ADSL

NYOMDAI MUNKÁK

DOMAIN REGISZTRÁCIÓ

MULTIMÉDIA

VÁLLALATI HÁLÓZATOK

15% kedvezmény a Codex olvasóinak*

WEBHOSTING

* A 15%-os kedvezmény webhosting szolgáltatásunkra érvényes, kizárólag a fenti hirdetés felmutatásával vehető igénybe.

Next IT

NEXT-IT Hungary Informatikai Kft.
1625 Budapest Pf.: 77 Telefon/fax: (+36-1) 405-6000
www.next-it.hu Email: info@next-it.hu

CODEX
Nemzetközi
Számítástechnikai-Programozói Szaklap

FŐSZERKESZTŐ:
Hevesi Zsolt
h.zsolt@codexonline.hu

SZERKESZTŐ
Dr. Medzihradszky Dénes

FELELŐS KIADÓ
CODEX INTERSOFT Bt.

GRAFIKUS
Kostean Norbert

TÖRDELŐ:
Kostean Norbert

FŐ TÁMOGATÓINK:
Oktatási Minisztérium
Márton Áron Szakkollégium

FŐ MÉDIATÁMOGATÓNK:
IDG Magyarország

SZAKMAI PARTNEREINK:
BioDigit Kft.
Next IT Hungary Kft.
Számalk Rt.

A **CODEX** INTERNETES ELÉRHETŐSÉGE:
www.codexonline.hu

NYOMDA:
Révai Nyomda Kft.
1037 Bp., Kunigunda útja 68

PÉLDÁNYSZÁM:
20.000

TERJESZTÉS:
Magyarországon és 7 határmenti országban

HU ISSN 1786-1136

Kedves Kolléga!

Ha érdekli Önt az informatikán belül a **programozás**, a számítástechnikai rendszerek működése, akkor ez a folyóirat Önnek szól!

Szeretnénk megismertetni egy új lehetőséget, melynek segítségével továbbfejlesztheti tudását, de ugyanakkor át is adhatja ismereteit másoknak. Egy kiterjedt oktatási projektről van szó, mely **"hetedhét országra szól".**

Elvünk, hogy a tudás terjesztésében nincs korhatárra vagy tudásszintre vonatkozó megkötés! A projekt keretében a hangsúlyt egy nemzetközi szemléletű, mind Internetes, mind nyomtatott formában megjelenő, az Olvasó **által is írható** folyóiratra helyezzük.

A **CodeX folyóirat** egy éven át folyamatosan megjelent az Interneten. Az elmúlt időszakban 10 szám került ki a világhálóra, melyekben összesen több mint 220 informatikai tárgyú cikk látott napvilágot. A népszerű témáktól kezdve - sokszor középiskolások, és felsőoktatási intézmények hallgatóinak tollából - egészen az új technológiákat bemutató szakcikkekig sok minden megtalálható.

Cikkeink **főként programozás**sal kapcsolatos témákat tárgyalnak, az **egyszerű szakmai fogásoktól a grafikai programok készítéséig, karakteres képernyőtől a WAP oldalakig**, különböző nyelvek és eszközök használatával. Megtalálhatók közöttük hardveres problémák leírásai és adatbázis kezelési ismeretek is.

Profiljának megfelelően a lap lehetőséget ad különböző témák publikálására; legyen az akár programozási feladat, vagy oktató, ismeretterjesztő jellegű cikk leközlése. Ez nem csak felajánlás, hanem kérés is, hiszen **mindenki ezreket taníthat a saját munkájával**. Aki pedig egy-egy programozási nyelvnek szakembere, sokat segíthet tudásával mint szaklektor a publikálni vágyó, a szakmában fiatal tehetségeknek.

E mostani nyomtatott folyóirat a sorozat 11. száma, hiszen az elektronikus lap folytatása. Így az egyes cikksorozatok itt kerülnek folytatásra. De természetesen új cikkekkel is jelentkezünk szokásunkhoz híven.

Hevesi Zsolt

KÖSZÖNETNYILVÁNÍTÁS
Köszönetet szeretnék mondani mind a 20 000 leendő Olvasónk nevében az Oktatási Minisztériumnak és a Márton Áron Szakkollégiumnak! Köszönetet a lehetőségért, mellyel a CodeX projektet kirobbanásszerűen nemzetközi viszonylatban is megismerhetik. A nyomtatott folyóirat alapul szolgál az informatikai szakma tanításában, a tehetségesek felkutatásában és nem utolsósorban a kapcsolatépítésben. És hálás vagyok, hogy nem hagyták veszni kemény hároméves munkám, és figyeltek az eddigi több, mint 12 000 egyedi online olvasónk érdeklődésére.

TartaloM

Eddig megjelent 10 számunk rövid összefoglalója

Eddigi cikkeink megtalálhatók a PC World 2004. szeptemberi CD mellékletén, illetve a CodeX hivatalos honlapján. Most, a már kiadott eddigi 10 szám tartalmából néhány érdekesebbet, fontosabbat megemlítünk:

1. SZÁM

Cikksorozataink:

ASP kezdőknek
WEB programozóknak az ASP programozás alapjairól.
A Delphi és a Win32 API
A Borland Delphi rendszerének hiányzó, vagy fel nem tüntetett komponenseivel, osztályaival foglalkozó sorozat. *Nem csak Delphi felhasználóknak!*
Vezérlés az LPT porton Alfától Omegáig
Számítógépünk csak akkor kezd el igazi életet élni, ha tudunk vele saját építésű perifériákat vezérelni. A cikksorozat az LPT portok egyszerű kezelésére vezet rá.
Java Szervletek
A webszerverek kiterjesztéseként működő „szervletek" nagyon hasznos feladatokra képesek.
Egy rendszergazda hétköznapjai...
Nem megszokott történetek, a rendszergazda mindennapi tevékenységei során átélt viszontagságos élményeiről.

Egyedi cikkeinkből:

Biztonság a weben
E cikk a Java védelmi rendszerén keresztül mutatja be a számítógépek és a hálózatok biztonságát.

Egyéb érdekességek:

Kliens-szerver architektúra; **Táblára fel**; **Felhasználóbarát alkalmazások készítése**; stb.

2. SZÁM

Új cikksorozat indul:

OpenGL.hu
A szerző megcélozza az érdeklődő grafikusokat, akik szeretnének jobban belemélyedni a játékkészítés és a 3D-s grafika rejtelmeibe.

Egyedi cikkek:

Egy DirectX váz
Szeretnénk megkönnyíteni az életét azoknak a grafikus programozóknak, akik Windows alatt programoznak.
Formás ablakok
Hogyan lehet megszabadulni a kötött formáktól? És szép vonalú grafikákkal, animációkkal tűzdelve látványossá tenni programjainkat?
Linux: Az SVGALib könyvtár
A Linux-al kapcsolatban elterjedt rémhíreket cáfolja meg ez a cikk, a Linux könnyebb, mint gondolnánk!

Folytatódnak sorozataink:

Dinamikus osztálybetöltés; **A Delphi és a Win32 API**; **Java Servletek**; **ASP Kezdőknek**; **Egy rendszergazda hétköznapjai...**; stb.

3. SZÁM

Legújabb cikksorozataink:

Írjunk böngészőt
A böngészőírás világába nyerhetünk betekintést.
A JDBC elmélete
A Java adatbázis-kezelési lehetősége korlátlan. A cikk a gyakorlatot alátámasztó elméletről ír.

Egyedi cikkeink:

JDBC gyakorlat
Az elméleti cikkhez kapcsolódó, kiegészítő cikk. A benne található forráskód mindenkinek a hasznára válhat.
A Velocity template Engine
A Velocity template engine egy sablon motor, melyet dinamikus weboldalak készítésénél tudunk használni.
DarkBasic PRO
A DarkBasic nyelv a játékprogramozás készítésében sokat segít; DirectX támogatásával egybekötve kiváló nyelv.

Egyéb érdekességek:

A Delphi szolgáltatásainak bővítése; Visual Basic Praktika; Adatbázis-kezelés Delphi-ben; **Az SQL lekérdező nyelv**; stb.

4. SZÁM

Új cikksorozataink

Az UDP kezelés Delphi alatt
Egy egyszerű, ám mégis praktikus üzenetküldő program készítéséről szól e cikk.
Hogyan írjunk saját Boot Managert?
Egy számítógépen több operációsrendszer használatához Boot Managerre van szükségünk. Ötleteket adunk, hogyan lehet saját Managert készíteni.
Objektumorientált programozás C++-ban
Osztály, mint alapegység. Az osztályok származtatása. Virtuális függvények

További cikkeink:

Pointerek "kicsit másképp"
Humoros megközelítése a pointerek sötét és zavaros világának.
Java objektumok leképezése relációs adatbázisokra Jakarta OJB-vel
Olyan haladó programozóknak nyújt segítséget, akik már tisztában vannak a Java nyelv lehetőségeivel.

Más érdekességek:

Java Server Pages; UML bemutató; stb.

5. SZÁM

Legújabb cikksorozataink:

Programozzunk Assemblyben
A cikksorozat célja, hogy fellebbentse a fátylat erről az alacsony szintű, elavultnak tartott nyelvről...
Bevezetés a mesterséges intelligenciába
A cikk célja, hogy figyelmünket az algoritmusok megtervezésére, a gépi gondolkodásra irányítsa.

További cikkek:

Jakarta Lucene keresőmotor
A Lucene egy nagyteljesítményű Java keresőmotor.
Letöltés-programozás
Az ismertetett eljárással könnyen és gyorsan megoldhatók a

5. SZÁM

Legújabb cikksorozataink:

Programozzunk Assemblyben
A cikksorozat célja, hogy fellebbentse a fátylat erről az alacsony szintű, elavultnak tartott nyelvről...
Bevezetés a mesterséges intelligenciába
A cikk célja, hogy figyelmünket az algoritmusok megtervezésére, a gépi gondolkodásra irányítsa.

További cikkek:

Jakarta Lucene keresőmotor
A Lucene egy nagyteljesítményű Java keresőmotor.
Letöltés-programozás
Az ismertetett eljárással könnyen és gyorsan megoldhatók a különböző állományok, dokumentumok letöltése.

További cikkeink:

Az adatok szűrésének lehetőségei; CGI programozása; pEdit 4.0 fejlesztői szövegszerkesztő; pBase 3.5 adatbázis-kezelő program; Táblaműveletek; stb.

6. SZÁM

Ujabb cikksorozatunk:

Bmp view
Képnézegető készítése Delphi segítségével.
Egyedi cikkeink:
Hogyan csináljunk BNC-t Delphi-ben?
A BNC feladata kizárólag annyi lesz, hogy egy megadott IP címre egy megadott port alapján felcsatlakozzon a célgépre...
Hogyan (ne) programozzunk?
Hogy ne „Spagetti" stílusban programozzunk...
Java naplózás megvalósítása Jakarta Log4J-vel
Olyan Java programozóknak nyújtunk segítséget, akik tisztában vannak a Java nyelv alapjaival, és egy megbízható, elterjedt, jól dokumentált naplózó rendszerre van szükségük.

Egyéb érdekességek:

Delphi Socket Komponensei; mIRC Script Programozás; stb.

7. SZÁM

Érdekességeink:

LPT Extra
Az LPT sorozat eddigi cikkeiből feltárult kérdésekre válaszul egy újabb cikk született, mely elárulja, miként lehet NT platform alól is elérni a portot ...
Többtáblás adatbázisok kezelése
Delphi segítségével kerülnek bemutatásra a többtáblás adatbázisok kezelése.
Delphi 6 Personal Edition
A Borland cég egy teljesen ingyenesen használható fejlesztőrendszert adott ki.
Hogyan válthatunk IP-t vagy host-ot az IRC-n?
Hogyan rejtsük el magunkat IRC-n?
Nyisztor Károly Gondolatok a programozásról
„A legjobb gondolataim pontosvesszőben végződnek."

8. SZÁM

Legújabb cikksorozatunk:

WAP-ra fel! Talán elgondolkodtunk már azon, hogy lehetne egy WAP-oldalunk, amit mobiltelefonról, PDA-ról stb. is elérhetnénk.

Figyelmet érdemlő cikkeink:

KeyBoard Hook
Programok írásánál gyakran szükségünk lehet a billentyűk figyelésére; esetleg egy adott kódrészlet lefuttatására.
USB eszközök kezelése
Windows XP-s megoldásra láthatunk egy példát.
Public, Protected, Private
A C illetve C ++ programozási nyelvek közti párhuzamot és különbséget tárgyaló cikk.

Egyéb érdekességek:

Assembly feladatok; Assembly direktívák, masm, link, regiszterek és még sok más; Java Proxy; stb.

9. SZÁM

Legújabb cikksorozataink:

Intel Assembler
A Turbo Assembler formai szabályaival, és a Turbo Pascal fejlesztőkörnyezethez való kapcsolódással ismerkedhetünk.
A registry kezelése Delphi-ben
A regiszterekben találhatók meg a Windows beállításainak többsége; remek eszköz lehet egy programozó kezében.
ViewNowX Server
A program a Unix/Linux szerverek X környezetét teszik hozzáférhetővé Windows rendszerek alól.
Játékfejlesztés
Játékfejlesztés rovatot indítottunk, reméljük, sokak örömére.
A következő témákat kiemelve:

- ✓ A modellezés alapkövei
- ✓ 3D program 4 sorban
- ✓ Moon Dragon interjú

10. SZÁM

Figyelmet érdemlő egyedi cikkeink:

Az Apache Tomcat és az Apache webkiszolgáló összeépítése
Úgynevezett szervlet konténer, amely lehetőséget ad Java nyelven írt szervletek JSP kipróbálására.
32bit-es Free Pascal grafika VGFX-el Windows alá
Azaz a VGFX grafikai egység bemutatása és alkalmazása.
Simson program
Ugyancsak a VGFX grafikai egység alkalmazása kerül előtérbe, de most egy konkrét gyakorlati megközelítésben.
Kapcsolat a külvilággal
A Runtime osztály feladata Java-ban; hogyan legyen kapcsolatunk a külvilággal?
Játékfejlesztés:

- ✓ Setup2Go
- ✓ 2D-s játék készítése

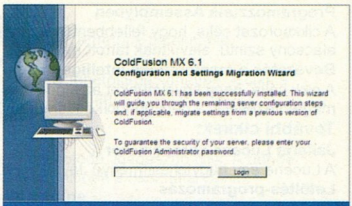
Sajnos, mostani számunk csak kevés cikket tartalmaz, de reméljük, elnyeri egyik-másik az olvasó tetszését!

TartaloM
+CD MELLÉKLET

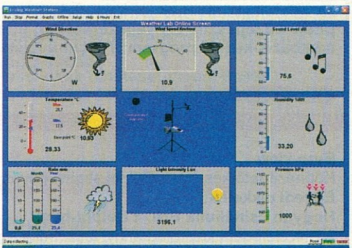
EXTRA CIKKÜNK:

.14 oldal

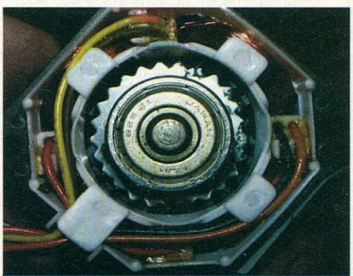
MAGYAR FEJLESZTÉSÜ BESZÉLŐ MOSÓGÉP -Szerzünk egy mosógépet kötött össze egy számítógéppel.



.10 oldal COLDFUSION MX 6.1



.22 oldal RENDSZERGAZDA HÉTKÖZNAPJAI



.20 oldal VEZÉRLÉS AZ LPT PORTON

CD MELLÉKLET

.CD BMP view - Képnézegető fejlesztése

Pályázati Felhívás!

A CODEX INTERSOFT Bt. nemzetközi programozói-cikkírói pályázatot hirdet

A CODEX INTERSOFT Bt. nemzetközi programozói-cikkírói pályázatot hirdet

"2004 legjobb CodeX cikkírója" címmel.

A pályázat célja, felkutatni a tehetséges cikkírókat, nemzetközi szinten, és publikálási lehetőséget biztosítani nekik. A dolgozat jellege szabadon választható, azaz eldönthető, hogy egy esszét ír, vagy egy programot mutat be. A téma és/vagy a téma feldolgozásában használatos programozási nyelv szabadon választható, de követelmény, hogy vagy egy programozási nyelvről szóljon, vagy egy feladat/példa (program) levezetéséről.

Két kategóriában indítjuk a megmérettetést a jelentkező felkészültségére való tekintettel: ***kezdő-haladó*** (közép- és felső-iskolás diák, fiatal pályakezdő), és ***szakképesített*** (több éve a szakmában tevékenykedő) megnevezésekkel. A két kategóriát összesen 550.000 Forinttal jutalmazzuk, de az elbírálás teljesen külön történik. Az első kategória keretösszege 200.000 Ft, a legjobb cikkeket 25.000-75.000 forinttal díjazzuk. A második kategória esetében a keretösszeg 350.000 forint, és a díjazottak 50.000-100.000 közötti összegben részesülnek. Mindkét kategóriában a nyertesek tárgyi ajándékokat is kaphatnak. A nyertesek listáját cikkeikkel együtt nyilvánosságra hozzuk.

A teljes keretösszeget, csak akkor osztjuk ki, ha megfelelő számú nyertes pályamunka született.

A nem díjazott, de jó értékelést elérő dolgozatok szerzőit lehetőségünk szerint megjutalmazzuk!

Az ötletes, találékony, jól felépített cikkeket díjazzuk, ahol a szakmai felkészülés és a témában való jártasság kimagaslik. Újszerű, frappáns megoldásokat várunk, melyek eddig nem, vagy csak részben szerepeltek a magyarországi publikációkban. A versenyzők hasonló eséllyel indulnak témától függetlenül, legyen szó hardver elemek direkt programozásáról, hálózati megoldásokról, operációs rendszerek programozásáról, webes alkalmazások fejlesztésekről, adatbázis kezelésről, animáció-készítésről vagy

A versenymunkák beadása folyamatos, de a végső határidő 2005 január 31. hétfő. A dolgozatok értékelése a novemberi hónap folyamán történik. A jelentkezést e-mail-ben kérjük, a cikkíró nevével, foglalkozásával, szakmai önéletrajzával, és a kész dolgozatával. Fontos továbbá feltüntetni a kategóriát, melyben indulni szeretne (első vagy második), illetve, ha van, akkor konzulensének nevét és titulusát. A két kategória, kortól függetlenül, indoklás nélkül szabadon választható.

A dolgozat 2-6 oldal A4-es terjedelmű lehet, 10-es betűmérettel és Times betűstílussal A dolgozatokat PDF formátumban várjuk (csak indokolt esetben küldjön HTML vagy WORD formátumot). Ha a dolgozat alapja egy elkészített program, akkor kérjük ennek futtatható változatát is (a működtetéséhez szükséges minden információval, figyelmeztetéssel stb. együtt). A teljes anyagot (cikket, példaprogramot, önéletrajzot stb.) egyszerre kérjük elküldeni, egy csatolt csomagolt formátumban (a csomagolt fájl mérete ne haladja meg az 1Mb-ot; zip, rar, ace formátumokat kérünk).

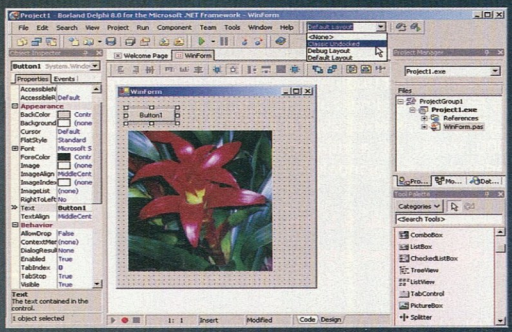
Bármilyen kérdése adódik, forduljon nyugodtan e-mail-ben a szerkesztőséghez: info@codexonline.hu

Komplex .NET-fejlesztés Delphi-ben

A fejlesztőeszközöket készítő Borland élete hagyományosan összefonódik a Pascal-al. A .NET Framework támogatására is megjelent Pascal alapú fejlesztőeszköz. Megjelenése a Borland új „környezetgazdálkodási" koncepciójának teljesebbé válásával időzítve zajlott.

Az ALM és a Delphi 8

Az új koncepció annak a felismerésnek a jegyében alakult ki, hogy a fejlesztői projektméretek esetében egyre jobban látszik az a tendencia, hogy nem az adott alkalmazás tényleges kódolása a legszűkebb keresztmetszet. Sőt, ha egy alkalmazás teljes életciklusát tekintjük, a fejlesztőeszközökbe épített automatizmusok okán, egy egyre kisebb szeletet jelent a tortából. Persze azt a legegyszerűbb esetben is láthatjuk, hogy egy program elkészültekor célszerű először elgondolni, hogy mit szeretnénk megvalósítani, majd azt végiggondolnunk, hogy hogyan szeretnénk megvalósítani, mielőtt nekiállnánk a spagettikód elkészítésének. Ahogy a megvalósítandó kódok, programrendszerek növekednek, úgy nő a szerepe az előzetes tervezésnek, az egyes elemek újrafelhasználhatóságának, a programkód és -működés dokumentációjának, az akár különböző programozók által készített elemek együttműködésének, a teljes rendszer optimalizálásának. Érthető tehát, hogy ezen feladatok ellátására számos műhelyben több megoldást is megvalósítottak, melyek jelentős mértékben képesek megkönnyíteni az alkalmazásfejlesztők életét és hatékonyabbá tenni a programfejlesztésre fejét adó cég működését.



A Fejlesztőeszköz a már megszokott módon teszi lehetővé a gyors alkalmazásfejlesztést a .NET-világ számára is.

A Borland, mint hagyományosan a fejlesztőket eszközökkel ellátó cég a fejlesztőkörnyezetek egyfajta kiterjesztéseként dolgozta ki azt a koncepciót, ami már túlnyúlik az adott IDE-ben zajló folyamatokon. Ez az Alkalmazás Életciklus Menedzsment (ALM), mely átfogja a nagy alkalmazásrendszerek tervezésétől a kódoláson keresztül a telepítésig a teljes fejlesztési folyamatot. Lényegében egy modell alapú alkalmazáskezelést téve lehetővé, melynek során az alkalmazás, alkalma-zásrendszer modelljét grafikus eszkö-zökkel előbb készíthetjük el, mintsem akár egy programsort is leírnánk. Márpedig, mivel a grafikus tervezés lényegesen áttekinthetőbbé teszi a világot, nem véletlen, hogy az UML alapú eszközök népszerűek a fejlesztők körében. A Borland kínálatában, illetve a Delphi 8 világában ezt a modellezést szolgálja Borland Enterprise Core Objects (ECO) for Delphi 8, mely része az Architect csomagnak, ahogy ugyan-csak ebben a kiszerelésben találjuk a Borland Optimizeit Profiler for the Microsoft .NET Framework is a futó kód jobbítására. A Borland teljes ALM-palettáján pedig ott találjuk a CaliberRM-et, az InterBase-t, a StarTeam-et illetve a Together-t is. Az, hogy az Architect-csomagban találjuk meg a teljesebb készletet, az érthető akkor is, ha az árazástól eltekintünk, mivel ez az a csomag, mely azoknak a nagy rend-szereknek az elkészítésére szolgál, melyeknél ki is tudjuk használni. Ugyanakkor a Delphi 8 kipróbálásához szintén az Architect változata áll rendelkezésre Borland honlapján (http://borland.com/delphi_net/).

A .NET Framework-höz való alkalmazásfej-lesztést azonban, érthető módon, valamennyi kiszerelés alapból tudja.

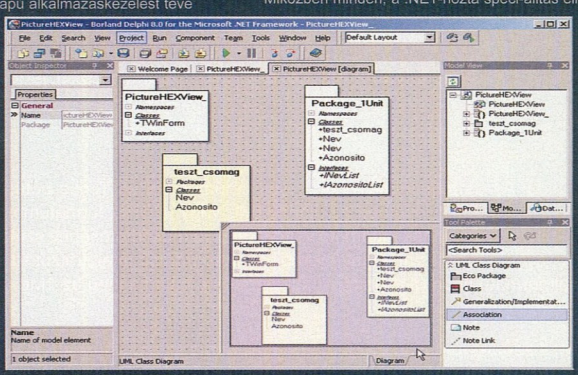
A fejlesztés

Mielőtt azonban a munkába fognánk, vessünk egy pillantást a Delphi 8 szolgáltatta munkakörnyezetre. Ez a C# Builder-rel dolgozók számára kevesebb újdonságot fog rejteni, míg a hagyományosabb Delphi-hez szokott felhasználóknak elsőre kicsit talán szokatlan lesz. Az alapkiépítés inkább a JBuilder-ben megszokottakra emlékeztet, de mert lehetőség van egyedi környezet kialakítására és elmentésére, ez aligha okoz majd gondot.

A .NET-es fejlesztés során a kódok rejthetnek meglepetéseket a korábbi tisztán Win32-es programfejlesztésekkel szemben. Nem is annyira globálisan, hanem a részletekben rejlő kisebb ördögök miatt. Sok szempontból, ha valaki például nem a Java irányából érkezik ebbe a világba, célszerű felidézni azokat a hagyományokat, melyeket a Turbo Pascal használa-latakor alakíthattunk ki. Annak idején bizony előfordulhatott, hogy azonos elnevezéssel előforduló eljárások fordultak elő a különböző UNIT-okban. Ilyenkor a fordítóprogram számára a UNIT.eljárásnév megadási forma tette egyértelművé, hogy mit is tegyen az adott kódrészlettel. A .NET-világban hasonlóan kezelhetjük a különböző névterületek (name space) objektumait, eljárásait, "alterületeit", vagy akár az egyes ilyen területeket a globális világhoz képest. Így érthető, ha a legegyszerűbb alkal-mazás elkészítésekor is ilyeneket olvashatunk a kód elején:

```
uses
SysUtils,System.Drawing,
System.Collections,
System.ComponentModel,
System.Windows.Forms, System.Data;
```

A névterületekről a segítségrendszer meglehetősen széleskörű ismertetést tartalmaz, de a .NET-környzettel foglalkozó más források is leírják ezeket. Ismeretük lényeges pontjává válhat a munkánknak, mert vannak esetek amikor sok bosszúságtól és hibaüzenetektől kímélhetjük meg magunkat ezen ismeretek birtokában. De a kód is követhetőbbé válik, ha követke-zetesen használjuk akár a változók megadásakor a .NET-es konvenciónak megfelelő formát, például "var color; System.Drawing.Color;" teljes név-területet feloldó formában. Ha további pontosításra van szükség, akkor pedig alkalmazhatjuk a "&" -jelet. Miközben minden, a .NET-hozta speci-alitás ellenére a hagyom-ányos Pascal, Object Pascal hagyományait követhetjük. Ha pedig valaki egészen "hagyományőrző" módon kíván fejleszteni, annak sincs akadálya. A Delphi 8 ugyanis támogatja nem csak a .NET Framework alapú FCL-komponenskönyvtárak, hanem a hagyományos VCL-komponensek hasz-nálatát is. Az ezeken alapuló programozásra a dobozból telepített eszköz mintaprogramgyűjteménye szintén ad példákat.



A Borland Enterprise Core Objects (ECO) for Delphi 8 segítségével modell-alapon tervezhetjük alkalmazásunkat.

Simay Endre

Rekurzivitás - elegáncia veszélyekkel

A függvényhívás - avagy mi történik a színfalak mögött
Függvényhívási konvenciók
Példaprogram - ParseDir könyvtárbejáró (C++ forráskóddal)

A következőkben a címben feltüntetett, nem kifejezetten mindennapi témákkal foglalkozom, és egy hasznos kis könyvtárbejáró program forrása is bemutatásra kerül, mely kiaknázza a rekurzivitás előnyeit.

Rekurzivitás - elegancia veszélyekkel

"Ahhoz, hogy megérthessük a rekurzivitást, előbb meg kell értsük a rekurzivitást."

Az előbbi mondatot egy C++ fórumon olvastam, és elég frappánsan foglalja össze a rekurzivitás lényegét.
A rekurzív szó legjobban az "önmagára hivatkozó", "önmagát hívó" fogalmakkal le-het megközelíteni, és az utóbbi forma az, ami legjobban tükrözi a jelentését, ha programozásról illetve függvényekről beszélünk. Az önmagát meghívó függvényt tehát re-kurzív függvénynek nevezzük.

```
void fv()
{
    // ... kód
    fv(); // a függvény ezen a ponton önmagát hívja
    // ... kód
}
```

A fenti kódrészletben az *fv()* fügvény implementációja látható, mely egy bizonyos ponton meghívja önmagát. Az a furcsa helyzet áll elő, hogy a hívó és a hívott függ-vény ugyanaz; lényeges, hogy a hívó függvény lokális változói ebben a különleges esetben is konzisztensek maradjanak.

Ahhoz, hogy megértsük a rekurzivitás eleganciája mögött megbúvó veszélyeket, nézzük meg, mi is történik valójában egy függvény hívásakor. Félretéve egy kis időre a rekurzivitást (de csak azért, hogy utána még nagyobb lendülettel rávessük magunkat), vizsgáljuk meg az alábbi kódot:

```
void g()
{
    //...
    int i = 0;
    short s = 255;

    fv();   // meghívjuk fv()-t

    //... fv() hívása után ide kerül a vezérlés

    return;
}
```

Amikor a *g()* függvényből meghívjuk *fv()* függvényt, gondoskodni kell *g()* lokális vál-tozóinak elmentéséről; ezek értéke adott esetben változatlan kell maradjon miután a vezérlés az *fv()* hívásának befejeztével újra a *g()* kódjában folytatódik. Szerencsére erről nem a programozó, hanem a fordító gondoskodik. Az ideiglenes illetve lokális változók tárolása az ún. verem-memóriában történik (angolul STACK), ami egy LIFO típusú adat-struktúra. A LIFO (Last In First Out) szó jól mutatja a verem szervezését: mindig az utoljára behelyezett elemet nyerjük ki először. A muködése nagyjából olyan, mintha tányérokat helyeznénk egymásra, és az utolsót tudjuk leemelni először (persze csak akkor, ha a megszokott módon közelítjük meg a problémát ;)).

Minden alkalmazás számára lefoglalásra kerül egy bizonyos méretu szelet a memóriá-ból, ennek egy része a verem-memória számára van fenntart-va. A veremmemória tehát véges, és "lefele nő", azaz az újabb elemek egyre csökkenő memóriacímekre kerülnek.

A függvényhívás - avagy mi történik a színfalak mögött

Most, hogy tisztáztuk a veremmemória fogalmát, megvizsgálhatjuk, mit is történik a kulisszák mögött egy függvény hívásakor.

"Egy függvény hívásakor a következő történik:
1.) a hívott *függvény paraméterei* bekerülnek a verembe (jobbról-balra haladva a paraméterlistában); A VisualC++ fordítója különböző függvényhívási kon-venciókat alkalmaz, és előfordulhat, hogy verem helyett a processzor re-isztereit használja, így optimalizálva a paraméterek átadását. A Microsoft-specifikus függvényhívási konvenciókra hamarosan kitérek.

Megjegyzés
Ha a függvényparaméterek érték szerint kerülnek átadásra, akkor létrejön róluk egy lokális másolat. Pointer vagy referencia átadása esetén ez nem történik meg, ezért osztály típusú paraméterek esetében megspórolhatjuk az ideiglenes objektum létrehozásával majd megszuntetésével járó - másoló konstruktor illetve destruktor - függvényhívásokat. A cím átadásával azonban lehetővé válik az objektum módosítása, amit kivédhetünk a const kulcsszóval.

2.) a *függvényhívást követő utasítás címe* is bekerül a verembe; ezáltal a pro-esszor tudni fogja, hogy a hívott függvényből visszatérve honnan kell foly-assa a végrehajtást

3.) a vezérlés a hívott függvény címére ugrik

4.) megtörténik a függvény saját *veremterületének* felállítása - létrejön az ún. *"stack-frame"*, amire a függvénynek a lokális illetve automatikus változói számára van szüksége

5.) az érték szerint átadott paraméterekből másolatok jönnek létre; ezek a másolatok lokálisak a hívott függvényben, ezért megszunnek a verem visszaejtése során;

6.) végrehajtódik a függvény kódja

7.) a függvény kódjának végrehajtása után megtörténik a függvény lokális ve-remterületének visszafejtése; ezt végezheti a hívó vagy a hívott függvény, függvényhívási konvenciótól függően. Lényeges, hogy a verem visszafejtés végén a verem utolsó eleme a 2. pontban elmentett *visszatérési cím* legyen - különben a vezérlés nem a megfelelő helyre kerül, aminek kellemetlen kö-vetkezményei lesznek. Ez szerencsére csak extrém esetekben fordulhat elő (lásd következő fejezet), hiszen a verem karbantartását a fordító által generált kód biztosítja."

"A függvényhívás. Függvényhívási konvenciók - mi történik a színfalak mögött"
részlet "Gyakorlati C++" c. könyvemből

A következő számban folytatom a rekurzivitás veszélyeinek valamint a megfelelő óvintézkedések bemutatását.

Addig is jó programozást és tiszta, hibátlan kódot kívánok! ;)

Nyisztor Károly
http://nkari.uw.hu

ColdFusion MX 1.rész

Dinamikus vagy statikus weboldalak

A dinamikus weboldalak valamilyen szerveroldali programozási nyelvet és adatbázist használnak, szemben a statikus oldalakkal, ahol előre rögzített tartalom jelenik meg a felhasználó előtt. Az utóbbiaknak nincs nagy jövője interaktivitási lehetőségek hiányában, 1-2 éven belül el fognak tűnni az Internetről.
Mit értünk interaktivitási lehetőség alatt? Például egy keresőmotort, egy szimpla szavazást, egy vendégkönyvet vagy éppen egy bonyolult fórumot. Itt jön képbe a Cold Fusion MX, mint fejlesztőeszköz, amivel a fentebb említetteket könnyedén megvalósíthatjuk.

Cold Fusion Bevezetés

A Cold Fusion MX egy webre szánt programozási nyelv. Ez egyáltalán nem egy fantasztikusan különleges nyelv, melyet kizárólag professzionális fejlesztőknek találtak volna ki. Ez az Macromedia terméke, kereskedelmi forgalomban érhető el. Világszerte több millió web-programozó és designer használja. A programnyelv már 6 éve van a piacon, és jelentős sikereket ért el. Ez azon nyelvek közé tartozik, ahol nagyon keveset kell programoznunk. Intelligens hibajavító kódja segítségével egy esetleges hiba bekövetkeztekor nem kell az egész algoritmust átfutnunk, egyből rámutat a hiba okára. A JSP (Sun), az ASP, az ASP.NET (Microsoft) és a PHP nyelvek szintén elterjedtek, de programozás terén közel sem olyan gyorsak és flexibilisek, mint a Cold Fusion MX. Egyetlen hátránya hogy pénzbe kerül. Két változatban érhető el: Cold Fusion MX Standard és Cold Fusion MX Enterprise. A Standard Windows és Linux alá, míg az Enterprise ezenkívül Solaris, HP-UX és AIX alá is elérhető. Számos adatbázist támogat (pl. MS Acces, ODBC, MS SQL Server, MySQL, PostgreSQL, Oracle, DB2, Informix, Sybase), a példákban a Microsoft Acces relációs adatbázist fogjuk használni.
A relációs adatbázis esetében az egyik adattáblát össze tudjuk kapcsolni egy másik táblával (nemcsak eggyel). Az adatbázis és a szerver-oldali nyelv közötti rést az un. SQL (Structured Query Language) nyelv köti össze. Az SQL utasításokat nemcsak a CFMX-ből érhetjük el, hanem egyéb más, adatbázisokat támogató szerver-oldali programozási nyelvekből. Az operációs rendszerek széles skáláját támogatja: Windows Server 2003 (IIS), Red Hat 7.3, 8.9, SuSE 8, Solaris 9, AIX 4.3.3, 5.1.

Publikálás az Internetre

A hazai tárhely-szolgáltatók közül jelenleg csak nagyon kevesen támogatják ezt a nyelvet és az árát is elkérik érte. Szerencsére akad egy-két ingyenes oldal, ahová feltölthetjük saját dolgainkat. Ilyen weboldal a www.cfm-resources.com. Regisztrálás után 30 napig próbálgathatjuk a rendszert.

Rendszerkövetelmények

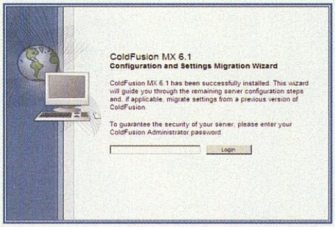
A Cold Fusion MX platformfüggetlen, tehát nem igényel speciális operációs rendszert. Megfelel neki egy Windows 98 / ME is, de a kisebb-nagyobb instabilitás miatt Windows 2000 SP3, Windows XP Professional vagy Windows Server 2003 ajánlott. Hardver terén egy 256 Mbyte-os memóriával, egy 300 Mhz-es processzorral és 400 Mbyte szabad tárhellyel rendelkező gépen gond nélkül fut, habár 512 Mbyte memória ajánlott.
Ahhoz, hogy futtatni is tudjuk alkalmazásainkat, szükségünk lesz az Internet Information Services (IIS) és a Cold Fusion MX 6.1 szerverre. A komplett fejlesztőkörnyezethez a Microsoft Acces és a Macromedia Dreamweaver MX is hozzátartozik.
Ha még nincs telepítve az IIS, akkor a Start menüben kattintsunk a Futtatásra, majd a következőt pötyögjük be: appwiz.cpl. Itt válasszuk ki a Windows összetevők hozzáadása vagy eltávolítása opciót. Az Internet Information Services-re lesz szükségünk, jelöljük ki és klikkeljünk a Továbbra. A telepítő kérni fogja a Windows install CD-jét.

Telepítés

Mikor idővel átálltam a Macromedia Cold Fusion 5-ről a Macromedia Cold Fusion MX 6.1-re (köztük még van egy verzió, a "sima" MX), gondban voltam a telepítésénél. Ugyanis szépen letöltöttem a 30 napos trialos telepítőjét (http://www.macromedia.com/cfusion/tdrc/index.cfm?product=coldfusion), majd pár OK és Next gomb hatására a program egyszerűen kilépett, bármilyen hibaüzenet nélkül. Hosszabb keresgélés után rátaláltam egy leírásra a www.macromedia.com oldalon, mely szerint a telepítési folyamatot az alábbiak szerint kell elvégezni (Windows alatt):

1. Hozzuk létre pl. C:\cftemp könyvtárt (fontos, hogy csak az angol ABC kis betűiből álljon!)
2. Sajátgép / Tulajdonságok / Rendszertulajdonságok / Környezeti változók -> a Temp és a Tmp változó értékét írjuk át a C:\cftemp-re, de előtte jegyezzük meg az eredetit, hogy a telepítés végeztével vissza tudjuk állítani
3. Kezdődhet az installálás
4. A végén kérni fog tőlünk egy jelszót, ezt feltétlenül őrizzük meg, mert majd csak ezzel érhetjük el az administrátort!
5.A művelet után állítsuk vissza a Temp és a Tmp értékét (legtöbb esetben ez a %USERPROFILE%\Local Settings\Temp) és töröljük a cftemp könyvtárt

Ezekre azért volt szükség, mert sajnos egyéb ilyen és olyan okok miatt nem kezeli a szóközös könyvtárakat a telepítő. Ha minden jól ment, akkor pár másodperc múlva megjelenik előttünk a Cold Fusion MX 6.1 konfigurációs varázslója, ahová a telepítésnél megadott jelszóval lehet belépni.



Az *RDS Setup* menüpontnál nem kötelező megadni jelszót, ha nincs megosztva a gépünk, akkor ez felesleges. Az *ODBC Setup*-nál egy kicsit várni kell, míg az adatforrásokat installálja. Választhatunk az *Example Application*-nál: szeretnénk-e példa alkalmazásokat vagy sem. Ha minden szükséges információt beállítottunk; megjelenik előttünk a CFMX 6.1 administrátora. Rendben, minden sikerült, ezt egy kis időre be is zárhatjuk.

Szintaktika

A CFMX funkciói a HTML-hez hasonlóan elemekből (tagekből) állnak, s így könnyen megtanulhatjuk, elsajátíthatjuk a nyelvet.
Itt is egy egyszerű példa:

form.html:
```
<form name="form" method="post" action="form.cfm">
<input name="nev" type="text" size="20">
<input name="tovabb" type="submit" value="Tovább">
</form>
```

form.cfm:
```
<cfoutput>
Szia <b>#nev#</b>, üdvözöllek a weboldalunkon!
</cfoutput>
```

A fájlokat a C:\Inetpub\wwwroot\demo könyvtárba mentsük el, ezek után a http://localhost/demo/form.html címen érhetjük el. Ha lefuttatjuk az oldalt, láthatjuk, hogy a <cfouput> elemmel írathatunk ki változókat a képernyőre. Ez bárhol szerepelhet a weboldalban, akár a weboldal kellős közepén is, és HTML kódokat is beágyazhatunk. A lap forrását nézegetve már csak a CFML fordító által legenerált kódot kapunk. A form.cfm így is kezdődhetne, ugyanazt az eredményt kapnánk:

```
Szia <b><cfoutput>#nev#</cfoutput></b>, üdvözöllek
weboldalunkon!
```

Változók használatával is megvalósíthatjuk a képernyőre való íratást:

```
<cfset valtozo='Hello Világ! '>
<cfoutput>#valtozo#</cfoutput>
```

Egy kis SQL

```
SELECT nev, email, telefon FROM tagok
```

A kód lefutáskor kilistázza a tagok neveit, e-mail címeit és telefon számait. Előtte persze rendelkeznünk kell egy adatbázissal, egy tagok táblával és a fentebb említett mezőkkel.

```
SELECT nev, email, telefon FROM tagok WHERE nev
LIKE 'a%'
```

A LIKE attribútummal kibővítve csak az "a" betűt tartalmazó neveket jeleníti meg.
Az adatbázisokat a Cold Fusion adminisztrátorában a ODBC Data Sources menüpont alatt kezelhetjük. Létrehozáskor ügyeljünk az adatbázis nevére: lehetőleg ne tartalmazzon szóközöket és kulcsszavakat (mint pl. a date vagy a text), mert ezek a későbbiekben csak hibához vezethetnek.

Adatbázisok kezelése
Készítettem egy példa adatbázist (http://www.programming.hu/samples/adatbazis.zip).
1. Letöltés és kicsomagolás után nyissuk meg az adminisztrátor (http://127.0.0.1/CFIDE/administrator/index.cfm)
2. A Data sources pont alatt vegyük fel az adatbázis nevét (adatbazis) és az adatbázis típusa a Microsoft Acces legyen
3. Adjuk meg az .mdb fájl útvonalát
4. Mentsük el és zárjuk be az adminisztrátort

C:\Inetpub\wwwroot\demo2:
index.cfm:
```
<cfquery datasource="adatbazis" name="lekerdezes">
SELECT * FROM felhasznalok ORDER BY nev ASC
</cfquery>
<table align="center" border="1">
<tr>
 <td><b>Név</b></td>
 <td><b>E-mail</b></td>
 <td><b>Telefon</b></td>
 <td><b>Azonosító</b></td>
</tr>
<cfoutput query="lekerdezes">
<tr>
 <td>#nev#</td>
 <td>#email#</td>
 <td>#telefon#</td>
 <td>#id#</td>
</tr>
</cfoutput>
</table>
```

A <cfquery> és a </cfquery> elemek között helyezkedik el maga az SQL kód. Két attribútumát specifikáltuk:
- datasource: itt az administrátorban felvett értéknek kell szerepelnie
- name: általunk meghatározott név, ez sem tartalmazhat szóközöket vagy írásjeleket

SQL magyarázat:
SELECT *: kiválasztja a táblában szereplő összes mezőt
FROM felhasznalok: a felhasznalok táblából
ORDER BY nev ASC: a találatokat növekvő ABC sorrendben jeleníti meg, a nev oszlop szerint. Az ASC helyett DESC-et is használhatnánk, ilyenkor csökkenő sorrendben jelenítené meg

Előzetes:

Röviden ennyi a Cold Fusion-ről. A cikk második részében bemutatom a CF-ben használt fontosabb függvényeket.
Forrákódok:
http://www.programming.hu/samples/demo.zip

Kiss-Tóth Marcel
marcell@programming.hu

Mit szólna egy új nyomtatóhoz?
... vagy egy TFT monitort szeretne ajándékba?

CodeX

Lehet, meg tudjuk Önt jutalmazni!

Keressen minél több éves előfizetőt !

☐■ - 8 darab egyéves előfizetés esetén – egy éves CodeX előfizetéssel jutalmazzuk, illetve 12 előfizető esetében egy újabb egy éves előfizetést adunk intézménye részére (könyvtárnak, munkatársnak)
- havi 12 megrendelt szám után (nem előfizetés) egy 13.-at küldünk ajándékba, illetve 16 példány mellé még kettőt

☐■■ 38 előfizető esetén Plextor CD írót adunk ajándékba

☐■■■ 50 darab egy éves előfizetéssel (vagy 10 hónapon keresztül havi 60 vásárló esetében):
HP tintasugaras nyomtatóval ajándékozzuk

... és kérdezi, hogy hány előfizetőt kellene összegyűjtenie ahhoz, hogy egy TFT monitort kaphasson? ... paraméterektől függően, de legalább 100 előfizetőt (vagy havi 120 vásárlót 10 hónapon keresztül).

Egyéb információ a www.codexonline.hu honlapon található.

Játékfejlesztés: percek alatt

Sok fiatal lelkes kezdő fejlesztő gondolja úgy, hogy egy játékot összedobni nem olyan nagy dolog... Részben igazuk is van. Vannak olyan fejlesztőeszközök, melyekkel viszonylag gyorsan össze tudunk barkácsolni egy játékot. Ebben az esetben egy FPS-t.

Sok kisebb, egyszerűbb nyelv létezik, melyekkel igen gyorsan érhetünk el látványos eredményeket. Ilyen például a DarkBasicPro. Most egy picike FPS játék alapjait írjuk meg. Percek alatt. Kezdésnek betöltünk egy pályát, majd beállítjuk az irányítást, hogy tudjunk nézelődni is. Vágjunk bele!

Első dolgunk, hogy beállítjuk a képernyőfrissítést. Megadhatnánk egy fix frissítési értéket is, de erre most nincs szükség. Ezután kikapcsoljuk az automata kamera módot. Ez azért kell, mert ha be lenne kapcsolva, akkor minden új betöltött objektum után megváltoztatná a kamera pozícióját. Beállítjuk a háttér színét a color backdrop rgb paranccsal, majd elrejtjük a kurzort a képernyőről a hide mouse paranccsal. Frissítünk egyet.

Most jöhet a kamera. Először beállítjuk, hogy hol helyezkedjen el a kamera, majd beállítjuk, hogy milyen távolságra lehessen ellátni a set camera range paranccsal.
Ezután betölthetjük a pályánkat. Mi most 2 fájlból állítjuk össze a pályát. Load object "fájlnév.kiterjesztés" objektum sorszáma Ezt tegyük meg mindkét objektummal. Ezután jöhet a ghost object on [objektum sorszáma]. Ez tulajdonképp az alfa elmosás. Beállítjuk az általános megvilágítás fényerősségét a set ambient light paranccsal. A 100-as érték jelenti a teljes megvilágítást, ahol nincsenek árnyékok. Most pedig helyezzük el a pálya részeit a position object paranccsal: position object objektum sorszáma, x, y, z koordináták. Ezután még méretezhetjük az objektumainkat.

A pályát betöltöttük, most jöjjön az irányítás:

```
if inkey$()="w" then move camera 1.0
if inkey$()="s" then move camera -0.5
if inkey$()="a" then turn camera left
1.0
if inkey$()="d" then turn camera right
1.0
```

Itt kedvünkre konfigurálhatjuk a billentyűzetkiosztást. A program leellenőrzi, hogy melyik billentyűt nyomtuk meg, és e programkód szerint fog eljárni. A kamera mozgatásánál az előre-hátra felé mozgáshoz move camera parancsot kell használni, oldalra történő mozgásnál pedig turn camera parancsot. De ahhoz, hogy mindez rendesen működjön, állandóan futnia kell, így ezt egy do - loop közé kell betennünk.
Ha gondoljuk, kiírathatjuk az FPS számot: print screen fps()
Frissítsünk egyet, majd jöhet a mehet! Ha minden igaz, valami ilyesmit kell látnod:

```
sync on
autocam off
color backdrop rgb(0,0,0)
hide mouse
sync

position camera 0,20,0
set camera range 10, 100000

sync

load object "codexlc.x",1
load object "codexlc_lm.x",2
ghost object on 2,1
set ambient light 70
position object 1,0,0,0
position object 2,0,0,0
scale object 1,10,10,10
scale object 2,10,10,10
do
if inkey$()="w" then move camera 1.0
if inkey$()="s" then move camera -0.5
if inkey$()="a" then turn camera left 1.0
if inkey$()="d" then turn camera right
1.0
print screen fps()
sync

loop
```

Ez indulásnak, kezdésnek elég. Van egy pályánk, amit be tudunk tölteni, mászkálni is tudunk rajta. Igen ám, csakhogy nem kell sok ahhoz, hogy észrevegyük: kimehetünk a nagy sötétségbe! Nem ütközünk a fallal.

A következő részekben megoldjuk, hogy ne tudjunk csak úgy kivándorolni a nagy semmibe, és még sok más dologra is megkeressük a választ.

De addig is:

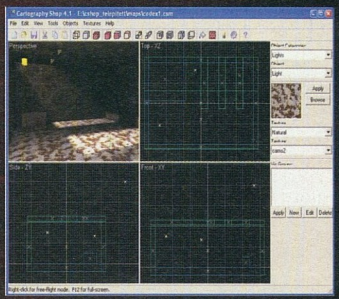
DarkBasicPro-hoz egyre több hasznos programot találhatunk a fejlesztő oldalán. Ilyen például a Cartography Shop, mellyel FPS-ekhez tudunk pályákat gyártani. A program támogatja .X és a .b3d, valamint a .map kiterjesztéseket (a teljes verzió). Így Nem csak DarkBasic-hez használhatjuk ezt a programocskát, hanem BlitzBasic 3D-hez is, valamint Half-Life pályák készítésére is jó. Ezen kívül még sok más prog-ramnak is használ vehetjük a fejlesztés során, és ismétlem, nem csak DarkBasic-es fejlesztők!

Összegzés:

Bár a fenti kép elég szép, már most tudjuk, hogy készülő játékunk nem fog vetekedni a Doom3-mal, de nem is ez a célunk. A lényeg, hogy relatív gyorsan fejlesszünk, és közben tanuljunk. Ez a sorozat pont ezt a célt szolgálja.

A következő részekben fejleszteni fogjuk az irányítást, a pályát, az ütközésvizsgálatot, és még sok minden mást is.

Tervezzünk...



Kapcsolódó oldalak:
www.thegamescreators.com
www.darkbasic.hu
www.jatekfejlesztes.hu

Keleti István jatekfejlesztes@jatekfejlesztes.hu

...és játsszunk!

mIRC vírusok eltávolítása

A mIRC elég fejlett script-nyelvvel rendelkezik. Sokan ellentmondanak ennek. Annyira mindenesetre fejlett, hogy a gépen bármihez hozzá tudjon férni a kliens program. Ez a vírusoknak is kedvez. Ha nem tudjuk, hogyan kellene használni, ha elhiszünk rossz embereknek mindenfélét, vagy ha egyszerűen csak nem vigyázunk, könnyen összeszedhetünk egy olyan vírust, ami a mIRC-en keresztül terjed, vagy fejti ki hatását.

Az itt leírtak tehát kifejezetten csak a mIRC alá írt vírusokra hatásosak, és csak a mIRC alatt működő részükre. Vírusokból sokféle létezik és a típusokat elválasztó sáv nagyon vékony, néhány esetben teljesen elmosódott. Sok esetben a vírusok több helyen is garázdálkodnak, többféle lehetőséget használnak fel a terjedéshez vagy a működéshez.
Minden esetben, ha egy-egy ilyen vírust leszedünk az itt leírtak segítségével, a gépünket nézzük át valamilyen víruskeresővel is, mert mint említettem, a vírusok sokfélék lehetnek, és megeshet, hogy az itteni ténykedésünk semmit nem ér, ha legközelebb a gép újraindításakor a vírus visszaírja magát!
Ha már nem volt szerencsénk, vagy elhittünk valamit, amit nem kellett vol-na, és elkaptunk egy vírust, akkor az eltávolítását többféleképpen tehetjük meg. Lépésről lépésre írom le, csak szépen követni kell a leírást, és ha valóban mIRC vírusba akadtunk bele, akkor le is fogjuk tudni szedni.

* Eltávolítás automatikusan, scripttel

Ez kényelmes megoldás, csupán a következő sort kell beírnunk a mIRC-be:

```
//while ($script(0)) { if ($exists($script(1)) == $true) { unload -rs $script(1) } | unsetall | timers off
```

Ez a következőket végzi el:
- az összes betöltött scriptet eltávolítja (csak eltávolítja, nem törli le!)
- az összes változót törli
- leállítja az időzítőket (timers)

vagyis visszaállítja a mIRC alapállapotát, olyan tiszta lesz, mintha éppen most tettük volna fel.

Sajnos néha a mIRC lefagy a végrehajtás közben (talán túl gyors a ciklus), de más gond nem lehet

* Eltávolítás mIRC-en belülről

Ez már nem olyan egyszerű, mint az első, de a hatása majdnem ugyanolyan, csak lassabb és biztosabb. Akkor használhatjuk, ha kicsit többet szeretnénk megtudni a mIRC működéséről vagy, hogy mit hol találhatunk meg benne. Szintén hasznos lehet, ha az előző módszerrel a mIRC-ünk lefagyott.

Hívjuk be a Remote szerkesztő ablakot. Ezt háromféleképpen tehetjük meg:
- menüből a `Tools/Remote`-ot kiválasztva (mIRC 6.1 előtt) vagy a `Tools/Scripts Editor`-t választva (mIRC 6.1 után)
- `Alt+R`-t nyomva
- az ikonját használva:

Így a mIRC beépített script-szerkesztőjébe kerülünk. Ez öt részből áll: `Aliases, Popups, Remote, Users, Variables`. Ezek közül nekünk a `Remote` kell, ha nem az lenne az aktuális, akkor válasszuk ki.
Ennek a szerkesztőnek van egy menüje is, nekünk egyelőre a `View` menüpont az érdekes. Itt vannak felsorolva a betöltött scriptek. Ha szerencsénk van, akkor csak egy darab `script.ini` van benne, és az is van kiválasztva.
Menjünk végig a scriptek listáján és válasszuk ki azokat egyesével. Ha kiválasztottunk egyet, akkor annak a tartalma megjelenik a szerkesztő részben.
Itt most kétféleképpen mehetünk tovább.
Az egyik módszer, hogy kiválasztjuk a következő menüpontot: `File/Unload`, amivel a megadott scriptet eltávolítja a mIRC-ből. A script állomány maga megmarad, csak a mIRC nem fogja használni.
A másik lehetőség, hogy amint a szerkesztőben megjelent a script, szépen kijelöljük az egészet (egérrel, vagy `Ctrl+A`), és megnyomjuk a `Del` gombot, vagyis kitöröljük!
Ha kitöröltük, akkor válasszuk ki a következő menüpontot: `File/Save` vagy nyomjuk meg a `Ctrl+S` billentyűket. Ezzel elmentettük a kiüritett scriptet. Így a script állománya megmarad ugyan, viszont teljesen üres lesz. Ezt mindegyik scripttel tegyük meg. Ha mindegyik már csak egy üres szerkesztőt hoz be; mikor újra kiválasztjuk, vagy a script listánk üres lesz, akkor tiszták vagyunk mIRC-script-ügyileg.

* Eltávolítás mIRC-en kívülről

Mondhatni ez a haladóknak szánt változat, ennél a változatnál a mIRC.INI állományával dolgozunk.
Ennél haladóbb már csak az lehet, ha kikeressük azokat a sorokat a scriptből, amik felelősek a vírus ténykedéséért, de ezt itt nem tárgyaljuk ;-)
Első lépésként csukjuk be a mIRC-et teljesen. Vagyis kapcsolódjunk szét a szervertől és zárjuk be a programot, vagyis ne fusson! Ha fut a mIRC-ünk, akkor nincs értelme az egésznek!

Aztán keressük meg, hova lett telepítve a mIRC-ünk. Ha nem tudjuk kideríteni, akkor a legjobb, ha jobb gombot nyomunk a mIRC ikonján vagy a menü bejegyzésén, kiválasztjuk a `Properties`-t (Tulajdonságok). Itt megtalálhatjuk, milyen útvonalról indul a program.
Ha megvan az útvonal, akkor odamegyünk. Megkeressük a `mirc.ini` állományt.

Valamilyen szerkesztővel vagy nézegetővel megnyitjuk. Megkeressük azt a részt benne, ami így kezdődik: `[rfiles]`

Ebben a részben vannak felsorolva azok a scriptek, amiket a mIRC betöltött. Ha semmilyen scriptet nem használunk, akkor valószínűleg csak a `remote.ini` (általában kétszer) és egy `script.ini` lehet benne, valahogyan így:

```
[rfiles]
n0=remote.ini
n1=remote.ini
n2=script.ini
```

Ha nem ezt találjuk, akkor máris kezdhetjük az ott felsorolt állományokkal. Ha csak a `remote.ini` és a script.ini van a listában, akkor belenézünk először a `script.ini`-be. Ha nem írtunk bele semmit mi magunk, akkor üresnek kell lennie. Ha mégsem az, akkor ebben az állományban található a vírus. Letöröljük ezt is. A `remote.ini`-ben nem szokott vírus lenni, de sosem lehet tudni, töröljük le azért ezt is.

Nyissuk meg a `mirc.ini`-t egy szerkesztővel, és a `[rfiles]` szekció alatti bejegyzéseket töröljük ki.
Mindezzel azt értük el, hogy a mIRC-ünk semmilyen scriptet nem tölt be induláskor. Ha a vírus ezek után is működik, akkor nem mIRC vírusról van szó.

Ha még durvábban szeretnénk eljárni, akkor a `[rfiles]` szekció törlése előtt megkeressük ezeket az állományokat egyenként. Szépen letöröljük azokat, szó nélkül. Ha nem használunk tudatosan scriptet, akkor nincs is rájuk szükségünk, valószínűleg azt sem tudjuk, mire valók, vagy mit csinálnak.

Egyszerűbben leírva: töröljünk minden olyan állományt, amit abban a listában találunk.
Ismét elindíthatjuk a mIRC-et és megint mehetünk IRC-re. Ha minden jól ment, akkor nem lehet több gondunk.

Balla Imre ircmania@talpak.org

PC ALAPÚ BESZÉLŐ MOSÓGÉP

Először- csak itt, a CODEX-ben!

A Magyar fejlesztésű beszélő mosógép bemutatkozik

Nemcsak a Japánok tudják megörvendeztetni a háziasszonyokat! Az LPT és Rendszergazda sorozatból régóta ismert szerzőnk egy érdekes berendezést épített, melyet 2004. június 12.-én, a GDF TDK - rendezvényén mutatott be, ahol a CODEX is jelen volt. Egy mosógépről van szó, mely a számítógépünkhöz kapcsolható. A mellékelt program elindítása után azonban megszólal;

"Jónapot kívánok, üdvözlöm a beszélő mosógép projekt kísérleti rendszerében. A folytatáshoz nyomja meg a T-t, a kilépéshez pedig a K-t!"

Első benyomása az lehet az átlagembernek, hogy az ALIEN című film képkockái elevenedtek meg az űrhajó beszélő számítógépéről. Sejtésünk csak erősödik, amikor munka közben a mosógép rendszeresen, előszóban informál bennünket, hogy mi történik pontosan a ruhákkal, s mennyi idő van hátra az adott műveletből... Mellékfunkcióként óránként bemondja a pontos időt is. A készítője ezt a készüléket fejlesztés alatt álló lakásvezérlő rendszere egy tagjaként tervezi a későbbiekben használni. Jelenleg azonban egy öreg notebook segítségével üzemeltetí. A hang "hátborzongatóan" tiszta, illetve a hozzákevert enyhe "kanyon-stílusú" visszhang miatt illik is a fürdőszoba hangulatához. A téma külön érdekessége, illetve aktualitása, hogy az LPT sorozatban leírt szilárdtest-relés motorvezérlő fokozatokat alkalmazta, minimális módosításokkal. Emellett a most induló "beszédeltáltás" cikksorozatában largyalt módon birja szóra a mosógépet. Szemléletes a példa: A szerző sok műhelytitokkal, programozási fogással ismertet meg bennünket, így ha figyelmesen olvassák lapunk megelőző, illetve elkövetkező számait, akkor az itt közölt információk, illetve a szoftverpéldák alapján bármely olvasónk képes lehet ilyen, vagy ennél is összetettebb feladatok megoldására. A szerző azonban -mint minden alkalommal-, ezúttal is óvatosságra int! Az erősáramhoz értő, érintésvédelemben jártos szakemberek segítségének igénybevétele létkérdés lehet a saját kivitelezésű berendezések esetén. Nélkülözhetetlen a biztonsági ellenőrzés, a felülvizsgálat is. S most ismerjük meg közelebbről is a beszélő mosógépet, - vagyis a "Mosómasát"- egyenesen a szerző tollából!

Mosómasa v1.00

Előzmények:

Igen régóta foglalkozom az intelligens lakás elkészítésének gondolatával. A nagy cégek jelenlegi megközelítésétől, illetve a divathóbortoktól, a luxustól eltérően egy olyan rendszert szeretnék létrehozni, mely a szociálisan hátrányos helyzetben lévő embertársaink életét könnyíti meg jelentős módon. A csökkent látóképességűek, vagy vakok mellett számtalan idős, vagy egyszerűen csak magányos, egyedül élő van, akiknek jólesne az emberi szó, még ha azt egy géptől kapják is meg. Ezért alapvető szempont, hogy a jövő berendezéseivel előszóban is lehessen kommunikálni. Talán hihetetlennek tűnik, de a technológia több, mint két évtizede megérett már rá. Azonban valamiért az ipar nem fejleszti kellőképpen e rendszereket. Furcsa módon, talán mi, fogyasztók vagyunk leginkább hibásak, mivel nem általánosan megfogalmazott elvárás egy elektromos eszköztől, hogy beszéljen, esetleg felismerje a szóban kiadott utasításokat. Úgy vélem azonban, valahol el kell kezdeni a fejlődést, illetve az áttörést. Ezért felvállaltam, hogy megtöröm a jeget, s szabadidőmben ilyeneket állítok elő. A lakásvezérlés azonban egy hatalmas projekt, ugyanakkor a hozzákapcsolható (elérhető áron, mindennapi kereskedelemben beszerezhető) háztartási, szórakoztatóelektronikai, illetve épületgépészeti berendezések a háztartásigép-iparban (egyelőre) tervezés alatt sincsenek. Még mindig autonóm rendszereket fejlesztenek, amiket 500-szor el lehet adni számunkra, minimális módosításokkal. Vannak természetesen kivételek, de az árak jóreszt csillagászati kategóriákat súrolnak. A fogyasztói társadalom pazarlása ebből adódik leginkább. Azonban lehetne ezt sokkal okosabban is végezni, miközben nem járna az ipar sem rosszabbul. A FÖLD ugyanakkor megmenekülne egy hatalmas adag ipari hulladéktól, a fogyasztók pedig korszerű és jól, tartósan használható, üzembiztos berendezésekhez juthatnának. Ma már nem okvetlenül szükséges olyan terméket előállítani, amik pár hónap alatt erkölcsileg, vagy fizikailag amortizálódnak; szétesnek, vagy elavulttá válnak. A tömegtermelést is lehet olcsón átállítani más feladatokra, ellentétben a kezdeti időkkel. Vannak tehát alternatívák, csupán meg kell őket keresni. Ennek jegyében láttam munkához. Két éve, beszélő légkompresszort építettem, most pedig egy régi, kidobásra szánt készüléket alakítottam át beszélő mosógéppé. Munkám során számos, hasznos tapasztalatot is szereztem, hiszen egy automata mosógép az egyik legösszetettebb tagnak számít a háztartási gépek családjában. Gyakorlatilag kicsiben szinte minden előfordul benne, ami egy lakásvezérlésben elképzelhető: fűtésszabályozás, folyadék feltöltése-leeresztése, szintszabályozása, elektromágneses szelepek, motorok vezérlése, s mindez egyszerre, a maga teljes komplexitásában. Ha valaki ezt egyszer megoldja, akkor nincs lehetetlen lakásvezérlési feladat, csupán kevés idő, vagy pénz a megvalósításra...

A mosógép átalakítása:

A gyári berendezés valamennyi vezetékét lebontottam, s teljesen új elektromos hálózatot építettem be. Ennek oka részben az eltérő működési mód, -részben- pedig az általam elégtelennek ítélt érintésvédelmi-biztonsági megoldások. A mosógép eredetileg kizárólag hidegvizes üzemre volt alkalmas. Ezzel szemben az átalakítás után a sokkal gazdaságosabb hideg-melegvizes mosásra is képes amellett, hogy a korábbi, hidegvizes táplálási lehetőség megmaradt. A régi muzeális felépítésű, de ma is számos új mosógépbe beépített elektromechanikus, gyakran meghibásodó, csörgőcsattogó programkapcsolót két nyomtatott panel váltotta fel. Az egyiken 10 db szilárdtest relé van elhelyezve, ezek végzik az erősáramú részek vezérlését, illetve a többi áramkörtől való biztonságos leválasztást. A másik, általam "vezérlőegységnek" nevezett panel az erősáramú fokozatot irányítja, fogadja a hőszenzorok, az ajtónyitás-érzékelő, valamint a vízmennyiség-szenzorok jeleit, s ezeket a szükséges biztonsági reteszelések után illeszti a számítógép nyomtatócsatlakozójához. A PC-n egy különleges program fut, mely a mosóprogram utasításainak, valamint a szenzorok jeleinek figyelembevételével jeleket ad a vezérlőpanelnek a beavatkozó szervekre vonatkozóan. A vezérlőpanel, a már említett módon be, illetve kikapcsolja az erőátviteli modulon lévő, megfelelő egységhez tartozó szilárdtest-reléket. A szenzorok állásáról, illetve a számítógép által kiadott parancsokról két, visszajelző-ledekből álló fényoszlopocska informálja a kezelőt. Ez a dekoratív külsőn kívül a saját, "cél-mosóprogramok" fejlesztésekor, módosításakor, vagy a berendezés javításakor

igen hasznos információkkal szolgálhat a gép kezelője számára.
Munkám során számos mérést, illetve kalibrációt, tesztet kellett elvégezni, melynek segítségével a mechanika, illetve a villamos berendezések működési sajátságait feltérképezhettem, s az optimális mosási paramétereket beállíthattam. Így végül egy olyan mosógépet kaptam - mely túl azon, hogy beszél -, valóban kiváló hatásfokkal, optimális víz, mosószer, illetve energiafelhasználással tisztítja a szöveteket. A végeredményt a 2. képcsoport baloldali fotója mutatja.

A beszélő szoftver:

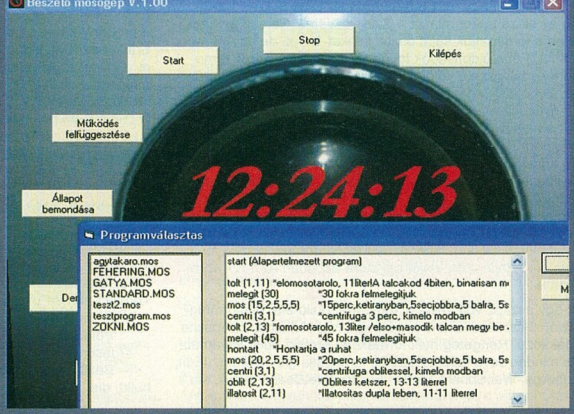
A mosógép - némi túlzással - "agyát" egy hozzákapcsolt számítógép adja. Ezen egy Visual Basic 6.0 nyelven megírt szoftver fut. A program voltaképpen önmagában nem tud mosni, ugyanis egy mosóinterpretert készítettem. Feladata csupán annyi, hogy a szöveges (TXT) formátumban megírt, 12-elemű utasításkészletből összerakott mosóprogramot lefuttassa. A programok .MOS kiterjesztésű állományok, melyeket be lehet tölteni, igény esetén módosítani, majd futtatni, leállítani, szüneteltetni, mint azt pl. a magnetofonok kezelőszervein is megszokhattuk. Emellett van egy információs gomb, aminek lenyomásakor a program tájékoztat a még hátralévő mosási időről. A mosóinterpreter a betöltéskor alapértelmezettnek a STANDARD.MOS állományt tekinti. Így ha betöltés nélkül, azonnal a startgombot nyomjuk meg, legkedvesebb mosóprogramunk indul rögtön. A kezelés tehát -a kevésbé hozzáértő családtagok számára- ezzel is egyszerűbb lehet. Mosóprogramokból már több mint 40-fajtát készítettem, de ezt mindenki magának, az adott feladatra optimálisan állíthatja össze. Ha egyszer egy program bevált, akkor egy rá jellemző néven elég csak elmenteni, s utána mindig megtaláljuk a "mosókönyvtárunkban". Az Interneten -amennyiben a berendezés népszerűvé válik-, mailben, FTP-n, WEB-en egyaránt lehet a hölgyeknek a bevált mosóprogramokat csereberélni, mint valami ételreceptet. Ezt az egyszerű, hatékony utasításkészlet segíti, mely nagymértékű kompatibilitást teremthet az eltérő berendezések között is. Eddigi tapasztalatom szerint egy alapvetően új érzésvilág, illetve kezelési szabadság ilyen berendezésekkel dolgozni. A 2. képcsoport jobboldali képe a szoftvert mutatja, a mosóprogramok választása közben.

Mosás közben a gép folyamatosan, élőszóban bemondja, hogy éppen mit tesz. Mikor melegíti a vizet, mennyi a hőmérséklet, hány liter vizet vett magához, mikor mos, mikor öblít, centrifugázik, stb... A hangmintákat többszörösen optimalizáltam. A hangbank egyelőre -önkéntes vállalkozó bemondó hiján- csak a saját hangomat tartalmazza, azonban a későbbiekben tervezem, hogy telepítéskor, esetleg a használat során több férfi, illetve női hang kiválasztható legyen. Terveim szerint a NET-es szupportról is lesznek majd letölthető hangok hozzá...

Ez a rendszer önmagáért beszél...

Természetesen a mikrosütőtől a mosogatógépen, gyeplocsoló automatán keresztül, a fóliasátrakig, riasztókig, személyfelvonókig, stb. a rendszer minimális módosításokkal adaptálható. Célom, hogy ezt az interpreteres, vagyis nagymértékben az igényekre szabható kezelési szabadságot az iparban, valamint a felhasználók között népszerűvé tegyem. Vége a kezelhetetlen, teljesen elvarázsolt, (ahány szerkezet, annyiféle távirányító, illetve menürendszer) koncepciónak! Ezzel kapcsolatban több szabadalmi oltalmat is be kívánunk nyújtani, s jelenleg tárgyalások folynak gyártó céggekkel a berendezés sorozatgyártását, illetve célszerű továbbfejlesztéseit illetően. Reményeim szerint ez a rendszer hamarosan a boltok polcaira, s végül a háziasszonyok kezébe kerülhet.
(A berendezés bemutató üzeméről egy rövid, nem egészen 8 perces MPEGII /és egy rövidebb MPEGI, amelyik verziója felfér/ video is készült, mely a röviditések, illetve kihagyások ellenére is jól szemlélteti a működést. A film letölthető a következő címről..... /Vagy a CD-mellékleten megtalálható/)

Kis Norbert - [redacted]



Apache Ant

Java-based build tool

Bevezetés

Bármilyen jellegű és bonyolultságú szoftver fejlesztésekor a programozó gyakran találkozik olyan műveletsorokkal, melyeket gyakran kell ismételni és részletesen paraméterezni. Ilyen lehet például egy fordítás, disztribúció készítése, telepítés, tesztelés.Egyszerűbb esetekben erre operációs rendszerhez kötődő scripteket használ, esetleg választ valamilyen gyakran használatos eszközt, mint pl. a make, gnumake, stb.
Java nyelvű fejlesztés során adott a programozási nyelv platformfüggetlensége, amit viszont megkötnek az előbbi eszközök operációs rendszerhez való szorosabb kötődése, illetve ezen eszközök konfigurációs fájljai sem a legbarátságosabbak.
Szerencsére a Java környezetben is szinte kvázi szabványná vált egy nyílt forrású eszköz, az Apache Ant (http://ant.apache.org). Írói kitűnően ötvözték a Java nyelv platformfüggetlenségét, az XML gép és ember által is könnyű olvashatóságát, illetve a kibővíthetőséget. Így egy kiváló, tiszta, XML konfiguráció által vezérelhető, Java osztályokkal bővíthető tool jött létre, melyet egy Java programozó sem nélkülözhet az eszköztárából. Ez a cikk az Ant alapjait mutatja meg, példákkal illusztrálva, javaslatokat téve a legoptimálisabb felhasználáshoz. Terjedelmi okok miatt a részletes bemutatás nem lehetséges, de a cikk tartalmaz utalásokat az Ant legtöbb képességére, az Olvasó dolga a konkrét megvalósításnak utánajárni.

Az Ant-ról

Az Ant-ot letölthetjük a hivatalos honlapról, a http://ant.apache.org/bindownload.cgi címről. A cikk írásakor a legutolsó verzió az 1.6.1-es verziószámot viseli, melyet 2004. február 12-én adtak ki. Letölthető bináris formában (zip, tar.gz, tar.bz2), illetve ha szükségünk van rá, forráskóddal együtt is. Ha valamilyen okból a legújabb verziót szeretnénk használni (pl. fejlesztés alatt lévő tulajdonság, vagy esetleges hiba), akkor letölthetjük a napi build-et is (Nightly Build), mely éjszakánként automatikusan generálódik, a CVS alapján.
Az Ant az Apache License 2.0 hatálya alá esik, azaz tulajdonképpen szabadon felhasználható, csak annyit követelnek meg tőlünk, hogy továbbterjesztéskor csatoljuk a licence-t az alkalmazásunk dokumentációjához.

Az Ant megismeréséhez a hivatalos honlapon jelentős mennyiségű dokumentáció áll rendelkezésünkre, egyrészt egy nagyon részletes dokumentáció, FAQ (gyakran feltett kérdések és válaszok), Wiki (dinamikusan fejlődő tudástár), illetve könyvajánlások, cikkek és prezentációk. Ha segítségre van szükségünk, nyugodtan fordulhatunk a levelezési listákhoz, illetve azok archívumához, valamint a JavaDoc alapú API dokumentációt és a forrást is böngészhetjük.

Jelentős különbség van az 1.5.4-es, illetve 1.6.0-ás verziók között. Az egyik legnagyobb változás, hogy az 1.6.0-ás Ant már csak a Java 1.2-vel kompatibilis, míg az előző támogatta az 1.1-es verziót is. Persze ez nem azt jelenti, hogy nem fordíthatjuk saját osztályainkat 1.1-el, de az Ant már csak az 1.2 felett fut. Főbb újdonságok továbbá: az XML névterek támogatása, mely megakadályozza a különböző saját fejlesztésű taszkok névütközését; új osztálybetöltő; adapterek; antlib, mely újrafelhasználható saját taszkok összegyűjtésére alkalmas. Ezeken kívül jelentős mennyiségű új funkció, taszk és hibajavítás is belekerült, melyekről bővebben a WHATSNEW nevű dokumentum tájékoztat, a változásokat egészen az 1.1-es verziótól nyomon követhetjük. A fejlesztők maximálisan ügyeltek arra, hogy az előző verzióval lehetőleg a legnagyobb mértékben kompatibilis maradjon, ne kelljen új build folyamatot, új XML fájlt írni. Ezért a verziószáma sem 2.0, mely jelzi, hogy nem egy teljesen új verzióról van szó. Az 1.6-os valószínűleg nem is az utolsó ebből a szériából, de már készül a 2.0 ás (CVS-ben Mutant és Myrmidion modul), mely az Ant teljes átírása, hogy gyorsabb, tisztább legyen. A fejlesztők idejét egyelőre azonban az 1.6.x sorozat fejlesztése köti le.

Kezdetben csak egy eszköz volt, mely a Tomcat-et build-elte platformfüggetlen módon, aztán átkerült a Jakarta projekt alá, majd felkerült az Apache top level projektjei közé. Rengeteg nyílt forráskódú projekt build folyamata épül rá, és szinte minden Java fejlesztőeszköz támogatja (pl. JBuilder, JEdit, JDeveloper, VisualAge, WebSphere Studio, Eclipse, NetBeans, IDEA, stb.).

Alapelemek

Az Ant-ot egy XML fájlal kell konfigurálni, mely egy projektet tartalmaz, illetve hozzá tartozóan legalább egy targetet. A targetek közül egyet ki kell jelölni alapértelmezettként. A target tulajdonképpen több tevékenységet (taszkot) összefoglaló egység. A taszkok a legkisebb egységek, ezek végzik a tulajdonképpeni műveleteket. A taszkok lehetnek beépített, opcionális, illetve akár saját fejlesztésű taszkok is. A projektek, targetek és taszkok mind XML tag-ekként (project, target, task) jelennek meg.

Project

A projektnek három tulajdonsága (XML tag tulajdonság) lehet, a neve (name), az alapértelmezett target neve (default) és a projekt alap könyvtára (basedir). Illetve lehet egy beágyazott description tagje, melyben szöveges leírást lehet megadni.

Target

Target lehet például a fordítás, ideiglenes fájlok törlése, fájlok előkészítése, deploy, disztribúció készítése, JavaDoc generálása, használati utasítás kiírása. Egy target függhet más target-ektől is (pl. a disztribúció készítése előtt fordítani kell), ezek neveit az adott target depends tulajdonságának értékeiként kell felsorolni, figyelembe véve a sorrendet. A target lefutását feltételhez is köthetjük, pontosabban egy property-hez. A property tulajdonképpen egy változó, mely vagy üres lehet, vagy szöveges értéket tárolhat. Ha azt akarjuk, hogy egy target csak akkor fusson le, ha egy property be van állítva, használjuk az if tulajdonságot, ha azt, hogy csak akkor fusson le, ha a property nincs beállítva, használjuk az unless tulajdonságot. Itt is megadhatunk szöveges leírást, de most már egy description tulajdonságként. Lehetőleg csak betűkből és számjegyekből álló nevet adjunk meg, kerüljük a szóköz és pont karaktereket. Ha egy target nevét kötőjellel kezdjük, akkor nem hívhatjuk meg a parancssorból.

Taszk

Egy taszk is egy XML tag-ként jelenik meg, és felparaméterezni a tag tulajdonságaival lehet. Minden taszkhoz lehet egyedi azonosítót rendelni, és scriptekből ezzel hivatkozni rá. Érdekes tulajdonság, hogy az Ant taszkokat nem szükségképpen csak Ant-ból lehet használni, hanem felhasználhatjuk őket saját programjainkban is, hiszen gyakran használatos funkciókat valósítanak meg, melyekre nekünk is szükségünk lehet.

Property

A property-k név és érték párok, ahol a név nagy- és kisbetű érzékeny. Értéket az XML fájlon belül és kívül is lehet adni. Hivatkozni egy propertyre a következőképpen lehet: ${property_neve}. Az Ant hozzáférést biztosít a rendszerjellemzőkhöz is (system properties) ezen property-ken keresztül, illetve van néhány beépített is (alap könyvtár, az XML fájl helye, Ant verziószáma, projekt neve, Ant-ot futtató Java verziószáma).

Példa

A következő példa egy egyszerű XML fájlt mutat be.

(1. példa Példa XML fájl)

```
<!-- Projekt nevű projekt, melynek az alapértelmezett target-je a dist, és az alap könyvtára
     az aktuális könyvtár. -->
<project name="projekt" default="dist" basedir=".">
   <description>
      Példa XML fájl
   </description>
  <!-- Beállít három property-t, melyek neve src, build, dist, értéke src, build, dist. -->
```

```
  <property name="src" location="src"/>
  <property name="build" location="build"/>
  <property name="dist"  location="dist"/>

  <!-- Init nevű target, mely létrehoz egy könyvtárat. -->
  <target name="init" description="Inicializáció">
    <!-- Létrehoz a build property-ben megadott névvel egy könyvtárat -
mkdir taszk. -->
    <mkdir dir="${build}"/>
  </target>

  <!--Compile nevű target, mely előtt le kell futtatni az init nevű target-et. --
>
  <target name="compile" depends="init"
       description="Fordítás " >
    <!--Lefordítja az src property-ben megadott könyvtárban lévő fájlokat,
és a class fájlokat
       a build property-ben megadott helyre teszi - javac taszk.-->
    <javac srcdir="${src}" destdir="${build}"/>
  </target>

  <!-- Dist nevű taszk, mely egy jar fájlt készít, és előtte le kell futtatni a
compile target-
       et. -->
  <target name="dist" depends="compile"
       description="JAR" >
    <!-- Létrehoz a build property-ben megadott névvel egy könyvtárat -
mkdir taszk. -->
    <mkdir dir="${dist}"/>

    <!-- Létrehoz egy JAR fájlt a dist property-ben megadott könyvtárba,
melybe a build
         property-ben megadott könyvtár tartalmát tömöríti be - jar taszk. -->
    <jar jarfile="${dist}/${ant.project.name}.jar" basedir="${build}"/>
  </target>

  <!-- Könyvtárak törlése target. -->
  <target name="clean" description="Törlés" >
    <!--Letörli a build és dist property-ben tárolt könyvtárakat. -->
    <delete dir="${build}"/>
    <delete dir="${dist}"/>
  </target>
</project>
```

Token filters

Az Ant lehetőséget biztosít arra, hogy fájlokban bizonyos szövegeket másokra cseréljünk le. Ez jól használható abban az esetben, ha egy közös fájlban szeretnénk eltárolni a különböző paramétereket, és csak telepítéskor behelyettesíteni az alkalmazás konfigurációs fájljaiba. A kicserélni kívánt szöveget token-nek nevezzük, és @token_nev@ formátumban kell a fájlban szerepeltetnünk. A név és az érték megfeleltetést a filter taszkban kell megadnunk, és a másolásnál szerepeltetni kell a filtering="on" tulajdonságot. Bináris fájlok másolásánál nem alkalmazható.

Path-típusú struktúrák

Abban az esetben, ha path-típusú struktúrákat kell megadnunk, beágyazott tag-eket kell alkalmaznunk. A fő tag lehet pl. a path vagy classpath, és az alá írhatunk újabb tag-eket, pl. pathelement, dirset, fileset és filelist.
A pathelement tag szerepelhet path tulajdonsággal, ezt akkor érdemes használni, ha az értéket egy property-ből kívánjuk beállítani, illetve szerepelhet a location tulajdonsággal, aminek értékeként egy, az alap könyvtárhoz relatív, vagy abszolút path-t kell megadni.
A dirset könyvtárak, a fileset fájlok csoportjai. Mindkettőnek lehet patternset altag-je, mellyel azt lehet definiálni, hogy az adott könyvtáron belül szűrünk-e bizonyos könyvtárakra vagy fájlokra. Ez tartalmaz include és exclude altag-eket, ahol az előbbi a nem szűrt, az utóbbi a szűrt könyvtárakat vagy csoportokat adja meg. Ezeket az altag-eket közvetlenül a dirset és fileset tag-ek alatt is lehet használni. Ezen egyszerű szűrési lehetőségeken felül lehetőség van úgynevezett szelektorok használatára, melyek különböző tulajdonságok alapján választják ki a könyvtárakat és fájlokat (pl. bizonyos karaktereket tartalmaz, dátum, más fájllal való egyezősége, méret, regexp, típus, módosítás alapján, stb.). Szelektorokat magunk is fejleszthetünk.
A filelist abban különbözik a fileset-től, hogy míg a fileset olyan fájlokat ad vissza, amik szerepelnek is a fájlrendszerben, a filelist-tel olyanokat is lehet definiálni, melyeknek szerepelniük kellene, de nem szerepelnek.

Abban az esetben, ha egy ilyen struktúrát több helyen is használni akarunk, akkor adjunk meg neki egy id tulajdonságot, és így máshonnan hivatkozhatunk rá. Megjegyzendő, hogy ez az összes olyan taszknál használható, amely beágyazott tag-eket is használ.

A következő példa a path-típusú struktúrákat mutatja be. Wildcard karakterek (csillag és kérdőjel) használata lehetséges.

(2. példa Path-típusú struktúrák)

```
<!--Path-típusú struktúra, azonosítóval. -->
<path id="base">
  <!-Property-vel meghatározott elem. -->
  <pathelement path="${classpath}"/>
  <!-Relatív elérési úttal meghatározott elem. -->
  <pathelement location="classes"/>
</path>
<!--Path-típusú struktúra. -->
<path>
  <!-- Hivatkozott elem -->
  <path refid="base"/>
  <!-- Könyvtárak -->
  <dirset dir="${build.dir}">
    <patternset>
      <include name="apps/**/classes"/>
      <exclude name="apps/**/*Test*"/>
    </patternset>
  </dirset>
  <!-- Fájlok -->
  <fileset dir="lib">
    <include name="**/*.jar"/>
  </fileset>
  <!-- Fájlok -->
  <fileset dir="lib">
    <include name="**/*.jar"/>
  </fileset>
  <!-- Fájlok -->
  <filelist
    id="docfiles"
    dir="${doc.src}"
    files="foo.xml,bar.xml"/>
</path>
```

Parancssori paraméterek

Parancssori paraméterek esetén az arg tag-et kell használni, ahol a value tulajdonság egy paramétert ad át, attól függetlenül, hogy van-e benne szóköz karakter. Ha a line tulajdonságot használjuk, akkor a szöveget a szóközöknél tagolja, és annyi paramétert ad át a taszk, ahány elem a tulajdonságban szerepel. A perjelek tetszőlegesen használhatók, az Ant automatikusan átfordítja az operációs rendszernek megfelelőre.

Az Ant futtatása

Az Ant a legtöbb elterjedt operációs rendszeren használható, úgymint Linux, üzleti Unix megvalósítások, mint Solaris és HP-UX, Windows 9x és NT, OS/2 Warp, Novell Netware 6 és MacOS X. Futtatásához és fordításához szükség van egy JDK 1.2-re, illetve egy XML feldolgozóra. A disztribúcióban az Apache Xerces2 szerepel, de ez tetszőlegesen kicserélhető. Installáláshoz elegendő kitömöríteni a letöltött tömörített fájlt; a bin könyvtárat elhelyezni a PATH környezeti változóban; beállítani az ANT_HOME környezeti változót, melynek arra a könyvtárra kell mutatnia, ahová az Ant-ot telepítettük; majd beállítani a JAVA_HOME környezeti változót, ez pedig arra a könyvtárra mutasson, ahol a JDK található.

Az Ant-ot futtatni egyszerűen az ant paranccsal kell. Ekkor az aktuális könyvtárban lévő build.xml-t használja, és futtatja az alapértelmezett target-jét. Az XML fájl nevét a buildfile kapcsolóval is meg lehet adni. Property-ket átadni a -Dproperty_neve=ertek kapcsolóval lehet. A parancs után meg lehet adni egy vagy több target nevet, melyeket le kell futtatni.

Bizonyos Ant taszkokat rendszerjellemzőkön át lehet beállítani. Mivel az Ant is Java nyelven készült, közvetlenül a java paranccsal is futtatni lehet.

FilterChain és FilterReader

A rövid UNIX parancsok és pipe nagyszerű használhatóságát próbálják az Ant-on belül a FilterChain és FilterReader tag-ek biztosítani.
Tulajdonképpen a copy tasznak lehet megadni egy FilterChain-t, mely FilterReader-ek egymásutánja. A FilterReader-ek a megadott sorrendben feldolgozzák a fájlt, és különböző beépített (pl. UNICODE escape-elés, a fájl első x sorának kiszűrése, grep, reguláris kifejezés, soronkénti prefix, csere, Java megjegyzések szűrése, sortörések szűrése, tab karakterek szóközzé konvertálása, tail, meghatározott karakterek törlése, összefűzés, tokenizálás) transzformációkat végeznek el rajta, de mi is fejleszthetünk ilyen FilterReader-t.

InputHandler

Az Ant a target futtatása közben maga is adatokat kérhet be a felhasználótól. Ez nem egyszerű konzolos bekérés, hiszen ekkor nem lehetne különböző IDE-kbe beilleszteni, hanem lehetőség van saját InputHandler megírására is. Az alapértelmezett InputHandler persze konzolról kéri be a felhasználótól az adatot, melyet Enter billenyűvel kell lezárnia.

Listener-ek és loggerek

Az Ant futásának figyelésére két módszer is lehetséges, egyrészt listenerekkel, melyek bizonyos esetekben (build indítása, build befejezése, target indítása, target befejezése, taszk indítása, taszk befejezése, üzenet loggolása) kerülnek meghívásra, másrészt logger-ekkel, melyek képesek elkapni a konzolra írt üzeneteket, és szűrni, formázni, továbbítani azokat.
Persze saját logger elkészítésére is lehetőségünk van.

Adapterek

Az adapterek Java osztályok, melyek olyan Java osztályokat alakítanak (pontosabban hívnak meg) Ant taszkká vagy típussá, amelyek nem terjesztik ki a megszabott interfészeket. Ezek tulajdonképpen olyan metataszkok, melyekkel dinamikusan lehet új taszkokat létrehozni.

Alap és opcionális taszkok

Az Ant ereje legfőképpen a szinte megszámlálhatatlan alap és opcionális taszkokban rejlik, illetve abban, hogy ilyeneket magunk is fejleszthetünk. Az előbbiek közül sorolok fel néhányat, a teljesség igénye nélkül.
Tömörítő taszkok alkalmasak Gzip, BZip, Cab, Zip (akár Jar, War, Ear), Rpm típusú fájlok előállítására és kitömörítésére, valamint Jar fájl aláírására.
Az Audit/Coverage taszkok alkalmasak a forrás fájlok felderítésére, analizálására, különböző mérőszámok kiszámolására.
A fordítást segítő taszkok alkalmasak Java forrás fájlok lefordítására, függőségek vizsgálatára, JSP fordításra, RMI csonkok generálására.
Egy deploy taszk segíti a J2EE környezetben szükséges alkalmazásszerverre történő telepítést.
Egy dokumentációs taszk a JavaDoc generálására alkalmas.
Külön EJB taszkok vannak, melyek különböző alkalmazásszerverek specifikus telepítésleíró fájljait képesek generálni. A következő alkalmazásszerverek támogatottak: Borland Application Server 4.5, iPlanet Application Server 6.0, JBoss 2.1 és későbbi verziók, Weblogic 4.5.1-től egészen az 7.0 EJB szerverekig, JOnAS 2.4.x és 2.5 nyílt forrású EJB szerver, valamint az IBM WebSphere 4.0.

OpenGL.Hu - X. Rész

Egy-két látványos apróság

Az utóbbi cikkben végre befejeztem a multitexturing bemutatását, ami már egy olyan technika, amelyet napjaink játékaiban is használnak. Az egyik leggyakoribb példa rá a lightmapping, aminek a lényege az, hogy egy megvilágított felületet úgy jelenítünk meg, hogy a fényt/árnyékot külön bitmapban tároljuk és multitextúrázással ráhúzzuk a megvilágítandó polygon-ra. Ez több okból is egy igen jól bevált és elterjedt megoldás. Azonban az OpenGL nem csak ilyen módon ad lehetőséget arra, hogy megvilágított/árnyékolt objektumokat jelenítsünk meg. A lightmapping nagy hátránya ugyanis, hogy minden árnyék, stb. statikus az objektumokhoz képest. A későbbiekben részletezni is fogjuk, hogyan oldható meg ez a probléma, hiszen a megfelelő megvilágítás elengedhetetlen eleme a 3D grafikának. Nélküle a jelenetek (elterjedt angol terminológiával: scene) elveszítik hitelességüket, szemetbántóan "mű" hatást kelt minden felület és anyag.

Most azonban vágjunk bele ennek a cikknek az anyagába! Az ehhez a cikkhez tartozó példa egy kicsit tömörre sikeredett, mert nem csak egyetlen témával foglalkozok benne. Az első mindjárt egy olyan dolog, aminek kihagyása eddig bizonyára sokakra zavaróan hatott.

Így teljes a világ!

Ez a rész arról szól, ami már minden grafikus felületű operációs rendszeren lassan alapkövetelmény egy kicsit komolyabb játéknál, vagy egyéb, 3D-s felületet használó programnál: a teljes képernyő használata. Az eddig elkövetett példaprogramoknál persze nem jelentkezett olyannyira a háttérben lévő menük, ablakok zavaró hatása, hiszen nem néztük ugyanazt a képernyőt órákon át, mint ahogyan azt akkor tesszük, amikor egy játékba merülünk bele. Ez ma már, mondhatni, alapkövetelmény. Lássuk tehát a kódot!

A módosítás elsősorban az inicializációs részt érinti, amely a főprogramban található. A függvény a korábbi példákban valahogy így nézett ki:

```
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
glutCreateWindow("Példa01");
glutDisplayFunc(RenderScene);
glutMainLoop();
return 0;
```

Ez a kódrészlet a 1. példaprogramból lett kivéve, így a minimális inicializáló utasításokat tartalmazza csak. Ebben a szituációban ez nekünk bőven elég, mert egyszerűbb lesz látni, mi változik.
A teljes képernyős módot a GLUT-ban igen találóan Game Mode-nak nevezték el, utalva arra, hogy főleg mire fogják a fejlesztők használni. A Game Mode inicializálása némileg eltér az ablakos módétól, hiszen nem kell például címsor, viszont szükség van a felbontás, színmélység direkt beállítására. Ez utóbbira eddig azért nem volt szükség mert grafikus felületű operációs rendszerek (továbbiakban: OS) esetén minden ablak ugyanazokat beállításokat annak létrejöttekor automatikusan lekérdezi le az OS-től.

Az új inicializáló rész valahogy így kell, hogy kinézzen:

```
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);

glutGameModeString("1024x768:32");
glutEnterGameMode();
glutDisplayFunc(RenderScene);
glutReshapeFunc(setupScreen);
glutIdleFunc(SceneIdle);

glutMainLoop();
return 0;
```

A glutGameModeString() és a glutEnterGameMode() függvények meghívásával történik a fenti beállítás. Az előbbi segítségével átadjuk a GLUT-nak azt, hogy milyen paraméterekkel szeretnénk a Game Mode-ot inicializálni. Az utóbbi függvény teszi meg ténylegesen a Game Mode bekapcsolását.
Megjegyzés szintjén hozzátenném, hogy a már korábban is említett M. J. Kilgard által írt referencia a glutFullScreen() használatát ajánlja erre a célra, azonban az a függvény nem teszi lehetővé a felbontás/színmélység beállítását, hanem az OS GUI-tól (az operációs rendszer grafikus felülete) lekért adatokat használja.

A glutReshapeFunc() függvény egy újabb finomítása az itt elsajátított technikáknak. Az eddigi példaprogramok nagy hibája volt, hogy ha a kerete megfogásával (dragging) átméreteztük az alkalmazásablakot, akkor elromlott a scene magasság/szélesség aránya. A glutReshapeFunc() alkalmazásával beállíthatunk egy függvényt, amelyik automatikusan meghívódik akkor, amikor az ablak átméreteződik (vagy Game Mode-ba lép!). Az itt beállított függvényben (setupScreen) érdemes elvégezni azokat a beállításokat, amelyeket befolyásolhat az ablak mérete. A setupScreen függvényünk kódja:

A futtató taszkok alkalmasak Ant futtatására, operációs rendszertől függő rendszerparancs hívására, JVM futtatására mind egymás után, mind párhuzamosan, ez utóbbi esetekben lehetőség van szinkronizálásra.
Külön taszkok vannak a fájlokon végzett műveletekre: attribútum állítása, checksum generálása és ellenőrzése, tulajdonos és jogosultságok megváltoztatása, másolása, törlése, fájl vége jelek konvertálása, mozgatás, könyvtár létrehozása, touch és patch parancsoknak megfelelő műveletek futtatása.
Ezen kívül lehetőség van loggoló beállítására, levél küldésére, üzenet kiírására, kulcs generálásra, hangfájl lejátszására, felhasználói interakcióra, SQL parancsok futtatására JDBC n keresztül, XML validálásra.
Számos preprocesszor taszk is rendelkezésünkre áll: pl. ANTLR, JavaCC, Javah, JJDoc, JJTree, XSLT transzformáció, valamint operációs rendszertől függő karakterek kicserélése escape-elt UNICODE karakterekre.
Lehetőség van buildnumber kezelésére, mely egy build-elésenként automatikusan növelt szám, property-k fájlból való betöltésére (properties vagy XML fájl).
Lehetőség van különböző hálózati protokollok és eszközök használatára: ftp, rexec, scp, ssh és telnet.
Támogat több változáskezelő rendszert is, úgymint CVS, Continuus/Synergy, Microsoft Visual SourceSafe, Perforce, Pvcs, SourceOffSite, StarTeam.
JUnit-tal való tesztelést is biztosítja.

Scriptelés

Az Ant egy különleges lehetősége a scriptelés, mely a BSF-en (Bean Scripting Framework) alapszik. A BSF Java osztályok gyűjteménye, melyek lehetővé teszik Java objektumok és metódusok elérését különböző script nyelvekből.Az Ant a JavaScript-et támogatja, melyhez a Rhino ECMAScript-et használja a Mozilla projektből. A JavaScript-et egy CDATA-ként kell megadni, és az összes Ant elem elérhető (pl. project, target, task, stb.) a nevével vagy egyedi azonosítójával (ID) hivatkozva.

Konklúzió

E cikk terjedelmi okok miatt csak egy rövid bepillantást nyújtott az Ant rejtelmeibe, felvillantva a lehetőségeket, hogy mit is nyújthat egy fejlesztés kapcsán ez a kvázi szabvánnyá vált eszköz. Tömörsége miatt lehetséges, hogy több hasznos szolgáltatás, illetve a taszkok pontos leírása is kimaradt. Ennek ellenére ezek megismerése nem nehéz feladat, hiszen az Ant elterjedése miatt rengeteg helyen lehet vele és róla szóló leírásokkal találkozni, és a hivatalos honlapon, illetve a letöltött disztribúcióban is jelentős mennyiségű dokumentáció fellelhető.

Hivatkozások

The Apache Software Foundation http://www.apache.org/
The Apache Ant Project http://ant.apache.org/

Viczián István
viczus@freemail.hu

```
void setupScreen(int w, int h)
  {
     if(h == 0) h = 1;
     glViewport(0,0,(GLsizei)w,(GLsizei)h) ;
     glMatrixMode(GL_PROJECTION) ;
     glLoadIdentity() ;
     gluPerspective(60.0,(GLfloat)w/(GLfloat)h,1.0,30.0) ;
     glMatrixMode(GL_MODELVIEW) ;
     glLoadIdentity() ;
  }
```

A függvény fejlécének formátuma kötött, ugyanis a megadott "h" és "w" paraméterpárban az ablak magasságát és szélességét kapjuk meg a híváskor. A glViewport() függvény ezt állítja be a ViewPort (a teljes ablakból illetve képernyőből látható felület) aktuális méretének, a gluPerspective() pedig ennek függvényében határozza meg a látóteret. Ez utóbbi függvény paraméterei sorrendben: y látószög, szélesség/magasság arány, legközelebbi pont, legtávolabbi pont. Az első sor értelemszerűen a jó öreg "Division by zero" hiba elkerülésére szolgál. Hát ennyi.

Ködösítsünk kicsit!

A Game Mode után most egy kissé más jellegű, ám annál látványosabb dolog következik. Emlékszem, még pár évvel ezelőtt, középiskolás koromban, mikor megjelentek az első DirectX-re épülő 3D-s játékok, meglepően gyorsan elharapózott egy igen könnyen kivitelezhető, ám annál látványosabb (és bizonyos esetekben hasznosabb) effektus használata. Ez az úgynevezett "distance fog" volt. Magyar kifejezést nem igazán használnak rá, esetleg a kicsit költői "távolság köde" illene rá a legjobban. (Megjegyzem, a későbbiekben egyre több olyan technika lesz itt tárgyalva, amely nevének nincs magyar fordítása, illetve amelyiknek van, annak nem általánosan elfogadott, erőltetett fordítása létezik csak.)

A dolog igazából elég egyszerű: ami távolabb van, azt a légkör miatt az emberi szem kicsit homályosabban látja, mint a közeli dolgokat. Szerencsénk van ezzel a technikával, mivel semmiféle extra trükk nem kell a használatához, az OpenGL-nek ugyanis megvannak a maga egyszerű függvényei erre a célra.

Most lássunk a szokásos módon némi kódot, aztán jöjjön a magyarázat. A példaprogramban az alábbi részhez hasonlót a RenderScene() függvényben találunk, esetünkben mást nem érint a distance fog használata.

```
GLfloat fogColor[4]= {0.7f, 0.7f, 0.7f, 1.0f};
glFogi(GL_FOG_MODE, GL_EXP2);
glFogfv(GL_FOG_COLOR, fogColor);
glFogf(GL_FOG_DENSITY, 0.5f);
glHint(GL_FOG_HINT, GL_DONT_CARE);
glFogf(GL_FOG_START, 8.0f);
glFogf(GL_FOG_END, 20.0f);
glEnable(GL_FOG);
```

A fogColor[] tömb értelemszerűen a szokásos 4f formátumú OpenGL színdefiníció, amely itt statikusan van megadva. Az utána lévő sor, amely a glFogi() függvény hívását tartalmazza, azt állítja be, hogy mennyire legyen részletes a ködünk. Ez igazából videokártya-teljesítmény kérdése, három lehetőségből lehet választani: GL_LINEAR, GL_EXP, GL_EXT2. Lehet vele szabadon próbálkozni, bár ha lehet, javaslom a GL_EXP2 használatát. Aki nem sejti miért, az próbálja ki! :
A továbbiakban a köd paramétereit állítjuk be:

GL_FOG_COLOR	A köd szine, egy vektort vár paraméterül,ezért glFogfv a függvény neve)
GL_FOG_DENSITY	A köd sűrűsége: 1.0f a legsűrűbb, 0.0f a legritkább
GL_FOG_HINT	A köd minőségét adja meg, lehet: GL_NICEST (szép, de elég lassú is lehet), GL_FASTEST (gyors, de nem annyira szép), illetve GL_DONT_CARE, ami talán a középút lenne.
GL_FOG_START	A kamerától mérve ennyi egységre kezdődik a köd...
GL_FOG_END	... és ennyire ér véget.

A már beállított ködöt a szokásos glEnable(), glDisable() utasításokkal kapcsolgathatjuk ki, illetve be.

Zárszóként megemlíteném, hogy a példaprogram egy hirtelen támadt ötlet gyümölcse: megpróbáltam egy kezdetleges módon megvalósítani az éjjeli/nappali fényviszonyok ciklikus változását a distance fog és némi szincserélgetés segítségével, úgy, hogy dinamikus fényt, vagy egyéb, bonyolultabb technikát még nem is használtam.

Mindenkit buzdítok, bátran kísérletezzen a kóddal, használva a fantáziáját!

Továbbra is várok mindenféle kommentárt, kérdést, melyekre ezúttal is szívesen válaszolok!

Merczel László - maniac@onlineweb.hu

Vezérlés az LPT-porton

Alfától Omegáig - XI. -rész

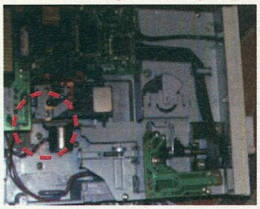
Az előző cikkünkben az iparban, háztartásban gyakran alkalmazott, 230V-os motorokkal barátkoztunk. Amennyiben erősebb motorokat is akartunk vezérelni, akkor az itt leírtaknak nagy hasznát vehettük. Most azonban ismét visszatérünk a törpefeszültségek világába; a léptetőmotorok vezérléséről lesz szó, melyekkel egyfokos tengelyelfordulásnál is finomabb lépésekben lehet forgatni a rákapcsolt mechanikákat, illetve nyilvántartani a pontos pozíciót... S mindezt tetemes forgató-nyomatékkal képes is megtenni! A léptetőmotor, - stepper motor -, voltaképpen a robottechnika, irányítástechnika, illetve a mai perifériák alapvető eleme. Legyen nyomtató, floppy, robotkar, CNC szerszámgép, plotter, szinte mindegyikben ezek vannak! A téma fontosságára, érdekességére, aktualitására való tekintettel ennek több részt is szentelünk, hogy a szükséges mértékben megismertethessük az olvasót ezzel a valóban csodálatos, illetve alapvetően fontos technológiával.

Léptetőmotorok I. - Vajon mi dorombol benne?

Hol találkozhatunk velük? Léptetőmotorok olyan berendezésekben vannak, ahol fontos, hogy a mechanika pozícióját pontosan ismerjük, illetve optimálisan változtatni tudjuk - nem túlzottan nagy sebesség, de igen kis tengelyelfordulási szögek mellett, viszonylag nagy nyomatékkal. Ilyen berendezések tipikusan a különféle floppy meghajtók, a régi, "batár méretű" (5,25-ös, általában MFM) merevlemezegységek, vagy nyomtatók, plotterek, s hasonló berendezések. A boltban az olcsóbb stepperek ára is viszonylag magas, azonban a fenti lepukkant, vagy egyszerűen divatjamúlt berendezésekből ingyen kitermelhetjük őket. Emiatt is érdemes alaposan megismerkedni ezekkel az ötletes szerkezetekkel. (Az első képcsoport példaképpen néhány olyan készüléket mutat be, ahonnan a stepperek ingyen kitermelhetők...)

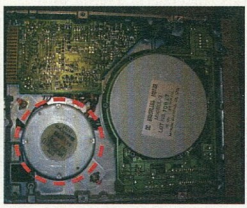
Ekkor egy mutatót nullázunk. Ezután minden előrelépésnél egy regisztert, vagy egy változót, - a "pozíciómutatót" - is növelünk eggyel, míg hátralépésnél csökkentünk. Így bármely pillanatban mutatónk értéke adja meg a pozíciót. Kis hátrány, hogy pl. egy áramszünet esetén akárhogy állhat motorunk, azaz bekapcsoláskor mindig alaphelyzetbe kell állnunk, illetve nulláznunk a mutatót, szakmai nyelven szólva "SEEK-elünk". Ez jól megfigyelhető a floppy meghajtóknál, nyomtatóknál, rajzgépeknél, robotkaroknál, stb. Amikor a gépet bekapcsoljuk, kis nyiszorgást, zümmögést hallunk; ez a hang ezt a folyamatot jelzi.
A lépéstévesztés fogalma: Megjegyzendő, hogy ennek a számlálásos technikának -mint mindennek az elektronikában- vannak gyenge pontjai. Pl, megeshet, hogy a mechanika megszorul; mert a motor lépni akart, de nem tudott. Ekkor elcsúszik a mutató,

1. ábracsoport - léptetőmotorok fellelhetősége



A régi, 1,2MB, illetve 360KB-os meghajtókban kocka, ritkábban henger alakú motorka mosolyog ránk békésen várva, hogy felébresszük...

A régi margarétakerekes nyomtatók fejében jókora motor megbújhat...



Másik floppyban tárcsa alakú stepper is lehet... A megbarnult címke ellenére a motornak semmi baja! :)

Régi, 5,25"-os, MFM merevlemezmeghajtókban is ilyen van, csak ehhez szét kell szedni!

A régi, 1,2MB, illetve 360KB-os meghajtókban kocka, ritkábban henger alakú motorka mosolyog ránk békésen várva, hogy felébresszük... A régi margarétakerekes nyomtatók fejében jókora motor megbújhat...Másik floppyban tárcsa alakú stepper is lehet... A megbarnult címke ellenére a motornak semmi baja! Régi, 5,25"-os, MFM merevlemezmeghajtókban is ilyen van, csak ehhez szét kell szedni!

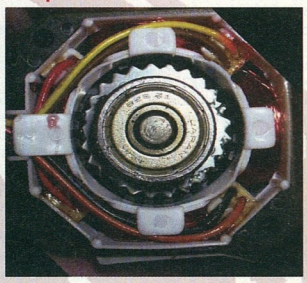
A léptetőmotorok feladata: A léptetőmotorok megszületését azon igény motiválta, hogy adott mechanikáknál igen pontosan kell a pozíciót szabályozni, illetve ismerni. Eközben az is fontos szempont, hogy a méregdrága pozíció-jeladók ne legyenek szükségesek. Ezért egy olyan motortípust kellett kifejleszteni, amit elektromos impulzusokkal "lökdösve", az egy-egy diszkrét pozíciót mozdul el. Így a kiadott villamos impulzusok számából a pozíciója is egyszerűen kiszámítható. Természetesen a helyzet mindig relatív: egy értékhez képest számítható. Ezért ezeket a motorokat végállás-érzékelővel szokták ellátni. Amikor a berendezést bekapcsoljuk, a motort addig forgatjuk, amíg a végállást el nem éri.

illetve a valós pozíció egymáshoz képest, vagyis "lépést téveszt" a motor. Ezért fontos, hogy a megfelelő feladatra mindig optimális teljesítményű (=forgatónyomatékú) motort alkalmazzuk. Ezenkívül menet közben időnként ellenőrizni is szoktuk, hogy nincs-e eltérés a pozícióban, ha valami kritikus művelet következik. (Amennyiben pl. az esztergakést ilyen motorral hajtva elkezdjük a forgó anyag felé tolni, nem árt ellenőrizni, nehogy késtörés legyen végül a dologból.) Vagyis menet közben is kell időnként SEEK-elnünk. A SEEK-ek között eltelt idő attól függ, mennyire terheljük meg a motort a teljesítményéhez képest, illetve milyen gyorsan forgatjuk. Nem szabad elfelejteni, hogy a mechanikának-a tömegéből adódóan- tehetetlensége van, így induláskor szeret "nem elindulni", mozgás végén pedig "tovább akar menni". Ilyenkor a motort érheti pár pillanatra a megengedett maximumnál nagyobb terhelés, ami tévesztést okoz. A helyes megoldás természetesen ilyenkor a fokozatos gyorsítás/lassítás. Azonban nem kell kétségbeesni; a tévesztés nem túl gyakori, (ha helyesen terveztük meg a berendezést) azonban kizárni sem lehet, így a szoftvernél mindig

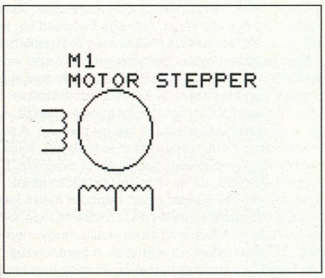
figyelembe kell vennünk!
Működési elv: A működést magát úgy lehet elképzelni, mintha egy tapsra egy katona egyet lépne, s minden lépés pontosan azonos hosszúságú lenne. Ha 5 lépésnyit akarok előre lépni, akkor ötöt tapsolok. Ha 10 lépésnyit, akkor 10-et. Az is fontos, hogy előre-hátra is tudjak lépkedni, vagyis másfajta tapsra visszafelé lépést kell megvalósítani.
A forgómotorok esetén jobbra-balra történő szögelfordulásról beszélhetünk, s lépésnagyság alatt az egy léptető jelre kapott elfordulási szöget értjük. Természetesen az már a mechanika kérdése, hogy ezt forgó, vagy egyenes vonalú mozgássá alakítjuk -e! A mi általunk alkalmazott motorok általában 1,8-fokos szögelfordulást végeznek, vagyis a teljes fordulatot 200 (200*1,8=360 fok) léptető impulzus hatására teszik meg. Az ember elsőre azt gondolná, hogy az ilyen motoroknak van egy előre forgató kivezetése, hátraforgató kivezetése, illetve közös szála, s ezzel a 3 vezetékkel tökéletesen lehet működtetni a szerkezetet. Sajnos azonban elektronika nélkül ilyen motort nehéz lenne egyszerűre, de ugyanakkor üzembiztosra elkészíteni. (Bár a telefontechnikában régen alkalmazott "marker-gépek" (keresőgépek, vagy léptetőjelfogók) egy irányba régen tudtak hasonlót, de azok nem voltak motorok, s nem is használhatók emiatt motorként.) A léptetőmotorok felépítését a 2. képcsoport szemlélteti:

2. ábracsoport - léptetőmotorok felépítése

Léptetőmotor szétszerelés előtt...



A forgórész, állórész mágneses szegmensei és a tekercsek előtűnnek a véglap eltávolítása után...



Elektromos jelölése rajzokon....Megfigyelhető, hogy két tekercs csoportban jelölik, de ez lehet a kiviteltől eltérő is. (bővebben a következő cikkben.)

Léptetőmotor szétszerelés előtt... A forgórész, állórész mágneses szegmensei és a tekercsek előtűnnek a véglap eltávolítása után… Elektromos jelölése rajzokon....Megfigyelhető, hogy két tekercs csoportban jelölik, de ez lehet a kiviteltől eltérő is. (bővebben a következő cikkben.)
A működés lényege, hogy a forgórész egy "fogaskerék" kiképzésű, erősen mágneses anyag. A fogak felmágnesezése felváltva északi-déli pólusokból áll. Vagyis, ha úgy tetszik, a fogaskerék mágneses változatával állunk szemben. Itt nem fizikai fogak, hanem mágneses mezők kapcsolódnak össze. Az állórész szintén ilyen fogakból áll, azonban a "fogak" (szaknyelven pólusok) mögött egy-egy tekercs foglal helyet. Ez hozhat létre mágneses teret, méghozzá szintén É-D sorrendben, váltakozva. Ha tehát ezeket a fogakat egyik irányban mágnesezzük fel, akkor a forgórészt egy foggal jobbra, ha másik irányban, akkor egy foggal balra igyekszik elfordítani. Természetesen egy 200-lépéses (1,8fokos) motorhoz 200 tekercset nem lehet beépíteni, mert nem férne el a drót az állórészben. Ezért aprócska "trükkökhöz" szoktak folyamodni a gyártók. Úgy alakítják ki a fogak számát, hogy nem kell se az álló, se a forgórészen 200 db belőlük. Ezt úgy érik el, hogy egyszerre nem az összes fog kapcsolódik, csak pl. minden negyedik (pl. a 4D-s motorok, amikkel mi is dolgozni fogunk). Ekkor -eddigi ismereteink szerint- 50 fogacska elég a forgórészen, az állón pedig néggyel kevesebb, vagy több. Ugyanakkor minden negyedik fogat ugyanaz a tekercs mágnesezi fel (a gyakorlatban elosztva van tekercselve, illetve a fogak mezőkre osztva, de az elv így sokkal szemléletesebb), tehát négy beépített tekercsünk van. A léptetéskor ezen tekercseket kell megfelelő sorrendben, egymás után áram alá helyezni, s a motor ekkor lép át a következő pozícióba. (A 3.-számú táblázat szemlélteti a ki/bekapcsolási sorrendet)

3. ábracsoport - léptetőmotorok vezérlési elve, egészlépés esetén (egyszerre csak egy tekercset kapcsolunk be)

1. lépésfázis	2. lépésfázis	3. lépésfázis	4. lépésfázis	5. lépésfázis	6. lépésfázis	7. lépésfázis	8. lépésfázis
Első tekercs be	Második tekercs be	Harmadik tekercs be	Negyedik tekercs be	Ismét első tekercs be	Második tekercs be	Harmadik tekercs be	S így tovább...

Megjegyzendő, hogy ennél picivel bonyolultabb a helyzet, mert a motor csupán féllépéses üzemmódban tudja az 1,8 fokos felbontást. (Erről részletesebben a következő cikkben lesz szó.) Így a fenti módon vezérelve mindössze 100 részre osztja csak a kört, vagyis egy lépés 3,6 fokos elfordulást eredményez, de a fogak száma is csak fele, azaz 25 a fenti 50-nek. Ennek csupán az egyszerűbb előállítás az oka.

A motoroknál tehát egyszerre egy tekercset áram alá helyezve szépen léptethetjük egyik pozícióból a másikba. A léptetési irány megfordul, ha a tekercseket fordított sorrendben gerjesztjük. Vagyis, ha 36 fokot (10*3,6fok) kell jobbra fordulni, akkor a következő a léptetési sorrend: 1,2,3,4,1,2,3,4,1,2. Ugyanez balra forgással: 4,3,2,1,4,3,2,1,4,3. Látható, ez igen precíz vezérlési megoldás, illetve a mai szoftver lehetőségek mellett igen egyszerűen megoldható feladat. A bemutatott léptetőmotorunk ennél sokkal többre képes, de erről már csak a következő folytatásban lesz szó… Addig is jó stepper gyűjtögetést kívánok mindenkinek! :::

Kis Norbert
norbimagan@freemail.hu

Egy rendszergazda hétköznapjai

A meteo és a makrók esete

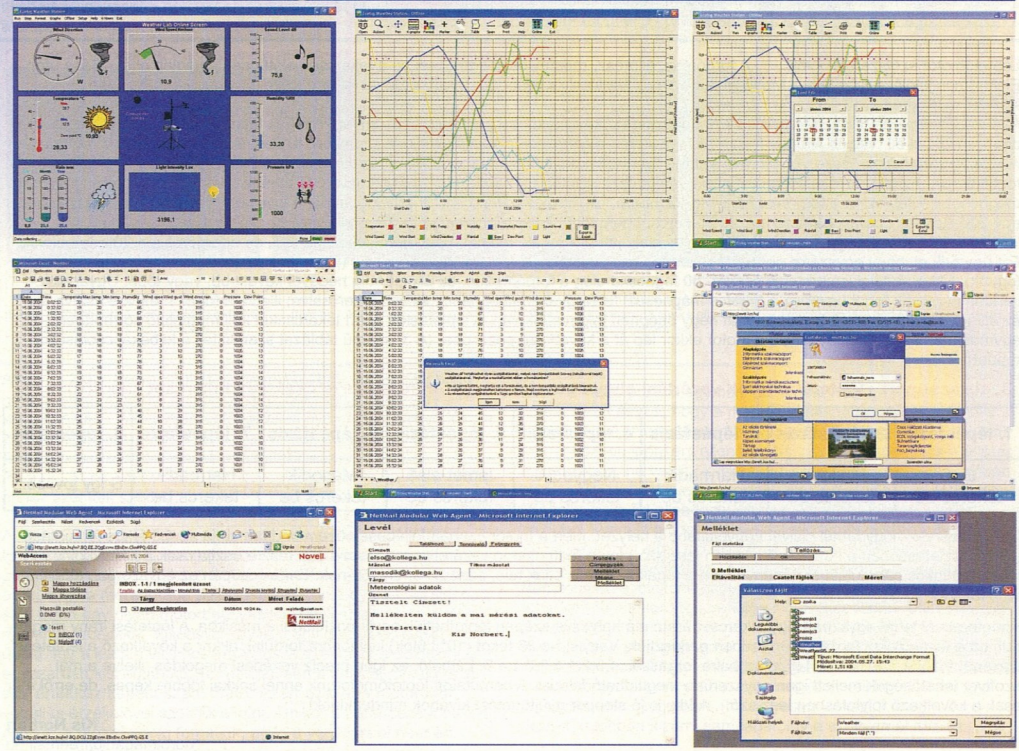
A rendszergazda munkája időnként érdekes problémákkal is szolgál. Ilyenkor a találékonyságán múlik, hogy kézzel, robotmunkát végez, vagy inkább rábízza a különféle automatizmusokra a feladatokat.... Ezúttal erről olvashatunk egy rövid, de annál tanulságosabb történetet.

A történet úgy esett, hogy a cég (suli) kapott egy meteorológiai adatgyűjtő állomást, amivel a tanár kollégák kísérleteket végezhetnek. Ez méri a hőmérsékletet, zajszintet, páratartalmat, légnyomást, csapadékot, a fényintenzitás változásait, a szél nagyságát, illetve irányát. A mérést egy szabadtéri egység végzi, ami egy beépített kis napkollektorról kapja az energiát, amivel a sötét órákra még akkumulátorban energiát is tartalékol. A mért adatokat másodpercenként kb. 2-szer egy beépített 423 Mhz-es rádióadón küldi le az aprócska vevőnek, ami egy tetszőleges PC RS232 portjára van kapcsolva. A PC-hez adott szoftver tárolja, illetve feldolgozza az adatokat. Táblázatokat, grafikonokat készít. A mérési ciklusok távolsága, a megjelenítés módja is testreszabható. A berendezést egy magas, használaton kívüli kéményre szereltük fel, illetve a rádiós megoldás helyett végül vezetéken hoztuk le az adatokat. Ennek a sztorija is szintén nagyon érdekes, de az már egy másik történet.

Azonban emellé feladat is jár: naponta mérni kell hétköznapokon az adatokat, s ezt eleinte Email-ben, később weben továbbítani a központi adatgyűjtő helyre. A feladatot komplikálja, hogy mérni minden nap, 14:00-ig kell, utána elküldeni az adatokat. A berendezés nagyon okos kis szerkezet, s a szoftverje is elég jól meg van írva. Azonban nem gondoltak a fejlesztők arra, hogy automatikusan küldhetők legyenek a mérési eredmények. Vagyis nincs benne egy SMTP-motor, -azaz olyan programrész,- ami Email-ben képes lenne adatokat elküldeni. Enélkül a kollégáknak szépen, naponta ismétlődő, rutinműveleteket kell elvégezniük. (A művelethez kb. 25 egérkattintás, 87 billentyűlenyomás lenne szükséges. Mindez, mint hamarosan megtudjuk: teljesen felesleges, mégis sokan így csinálják országszerte...)

Az ábrán a következő részműveletek láthatók: A folyamatosan futó adatgyűjtőprogramot a tálcáról teljes képernyőssé tesszük, hogy rendesen tudjunk vele dolgozni. Az egérrel (vagy billentyűkombinációval) aktiváljuk az OFFLINE menüpontot. (első ábra) Ezután megjelenik a második képen látható grafikon. Itt kiválasztjuk az intervallumot állító menüpontot, és a naptárban az aktuális naptól aktuális napig terjedő időszakot. (Ez egyszerű ENTER után automatikusan is beáll arra, nem kell kattogtatni ide-oda.) A harmadik képecskén ezt láthatjuk a felső sorban. Ezután (sajnos nincs billentyűkombinációja) az egérrel rá kell a fenti kép bal alsó sarkában található "EXPORT to EXCEL" gombra kattintani. Ez szépen elindítja az EXCEL-t, s beteszi a táblázatba a félnaphoz tartozó mérési adatokat.

1. ábra: a meteorológiai állomás kezelési lépései

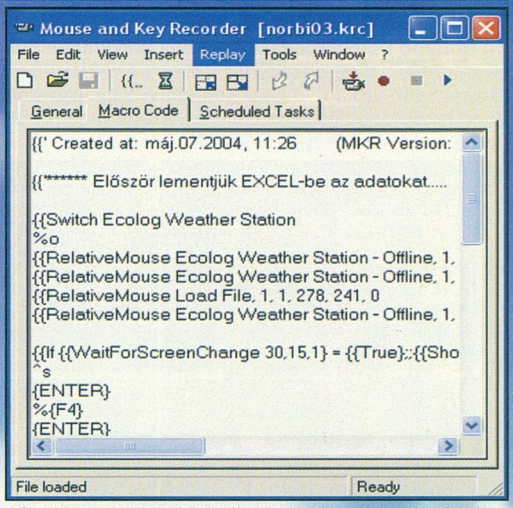


(második sor, első kép). Ezután jön a következő képen látható mentés művelet, amit célszerűen a CTRL+S kombinációval gyorsan elő lehet hívni. Igen ám, de a formátumra vonatkozó kérdéssel még kellemetlenkedik az Excel, majd arra is rákérdez, akarjuk -e a tegnapi mentést felülírni. Mivel a program adatbázisából úgyis bármikor előkaparható, természetesen igennel felelhetünk. Ezután ALT+F4-el bezárjuk az EXCEL-t, s még mindig nem örülünk, mert a mentett anyagot még levélben el is kell küldenünk a kollégáknak, akik szintén feldolgozzák az adatokat. Tehát a következő képeknek megfelelően elindítjuk a levelezőt, bejelentkezünk nevünkkel, jelszavunkkal, új levél írását kezdeményezzük, majd udvariasságból azért valami szöveget is írunk, amiben elmondjuk, mit küldünk. Végül a csatolás gombra kattintva a tallózó ablakban betallózzuk az előbb mentett adatokat, kérjük a levelezőt, hogy adja a csatolmányokhoz, majd elküldjük az email-t, kilépünk a levelezőből, s a meteorológiai programban visszalépünk online, vagyis alapértelmezett módba, majd lekapcsoljuk a monitort, s végre örülünk a jólvégzett munkának...

Főzzük meg Kolumbusz tojását!:

Talán mondanom se kell, hogy az ember egy hét után beleún az efféle értelmetlen dolgokba. Mi a csodának egyáltalán korunk csodája, a számítógép, ha ezeket az értelmetlen, monoton feladatokat naponta, kézzel kell megismételni? Eleinte arra gondoltunk, hogy írunk egy önálló feldolgozóprogramot, de az adatformátumokat sajnos nem kaptuk meg. Ki is lehetne persze okoskodni, de az sok idő. Valami frappáns, egyszerű, de nagyszerű, illetve általános megoldás kell a problémára... Ekkor jutott az eszembe, hogy régen volt egy makrórögzítő a windows alatt. Mi lenne, ha egy efféle programocskának elmutogatnám egyszer, mit kell csinálni, s utána ezt a programot valami elindítaná naponta egyszer, a megfelelő időben, (pl. a feladatütemező), s akkor az egylépésben megcsinálná a fentieket? Mondanom sem kell, hogy az ötletet tettek követték. A neten keresgélve pár perc alatt ráakadtam egy makró-rekorder programra. Érdekessége, hogy teljes értékűen használható 3 hónapig a DEMO verziója is. Lehet létrehozni, szövegszerkesztővel módosítani, majd lementeni makrókat. A makrókat azonban egy saját, egyszerű utasításkészletű nyelven programba is tehetjük. Vagyis lehet ciklusokat szervezni, feltételeket figyelni (pl. adott weblap betöltődött -e, adott ablak megnyílt, vagy aktív-e, stb...). Lehet időzíteni, billentyűleütéstől, beírt szövegtől, egérkattintástól függővé tenni komplett műveletsorokat, s mindezt alkalmazásoktól függetlenül, az operációs rendszert közvetlenül használva. A beépített makrórögzítő megjegyzi a lenyomott gombokat, egérkurzor pozíciókat. A rögzítést leállíthatjuk, vagy felfüggeszthetjük, s az így kapott szöveges formátumú információt beleszőhetjük makróprogramunkba. Ezen kívül mégegy hatalmas funkcióval rendelkezik: van egy időzítő menüpontja, ahol a windows ütemező (scheduler) kezelőfelületének összes funkciójával lehet egyszer, többször, naponta, hetente, hétköznap, hétvégén, stb. lefutó makrókat definiálni. Az aktivizálás után a makró meg is jelenik az operációs rendszer feladatütemezőjében, ahol szintén lehet módosítani az időzítéseken. No, ezzel a programmal az első képcsoporton lévő műveletsor mostmár automatikusan lefut, így emberi beavatkozás nélkül, precizen mindent megcsinál a számítógép. Ennek így kell működnie, de azért az emberben halkan felmerül a kérdés, amit tanulságként az összes szoftverírónak érdemes lenne megfogadni: egy szoftvert egyszer kell az életben megírni, de kezelésnél minduntalan végig kell menni minden marhaságon. Tehát kulcskérdés, hogy lehetőleg egyetlen felesleges gombnyomás ne legyen benne, hiszen ezt a felhasználók milliói egyenként sokezerszer kénytelenek megtenni. Ha ezt összeadjuk, akkor azt kapjuk, hogy a világ összes szoftverének a felesleges funkciója sokezer ember teljes életét kitölti. Értelmes, felnőtt emberek egész életükben csak a felesleges gombokat nyomkodhatnák, értelmetlen üzenetekre kattintgatnának. Kinek jó ez az értelmetlen pótcselekvés?

A makrórögzítő program kezelőfelülete:



A file menüpontban lehet betölteni/menteni a kész, programokba foglalt makrókat. Az edit menüpontban szerkesztő ablakot kapunk, amiben az utasításokat írhatjuk. Jó súgója van, könnyű eligazodni benne. Az insert menüpontban beszúrhatjuk programutasítás is. Az utasításokhoz példákat is generál, ha kérjük. Szintén ebben a pontban lehet makrókat rögzíteni. A tools-ban tudunk időzített indításokat beállítani. Számos funkciót megtalálhatunk a felső ikonsorban is, ez megkönnyíti életünket. Összességében elég barátságos kis segédprogram benyomását kelti a "Mouse and key recorder" névre hallgató szoftver. A neten számtalan hasonló szoftver van, ezért érdemes mindenképpen körülnézni, mielőtt egyik-másik mellett letesszük voksot.

(Akit érdekel a kis program, innen le is töltheti a 3 hónapig ingyen használható teljesértékű demo verzióját. http://www.softpedia.com/public/cat/13/9/13-9-153.shtml A készített makrót a CODEX webportáljáról letölthetik azok, akiknek szintén ugyanezt a feladatot kell végezniük naponta, vagy csak egy gyakorlati példára kíváncsiak.)

Kis Norbert - norbimagan@freemail.hu

Delphi és Win32 API

Nem csak Delphi felhasználóknak: X. rész

Windows NT eseménynapló - 1. rész

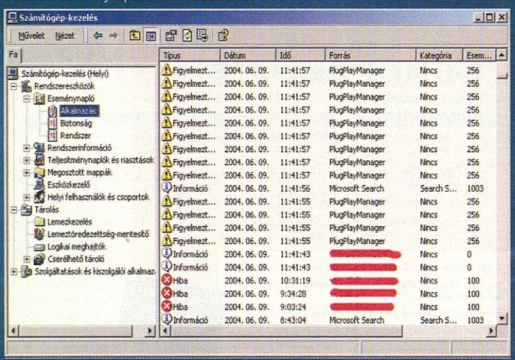
A Borland Delphi rendszer számos, a Windowsban (és a Linuxban) használt rendszereszközhöz tartalmaz komponenseket, osztályokat. Mégis érthetetlen módon több olyan elem hiányzik belőle, melyek sokszor vagy csak ritkán, de jól jönnének egy program fejlesztése során.Cikksorozatunkban ilyen komponenseket fogunk készíteni. A sorozatot más programnyelvek használóinak is ajánlom, hiszen a be-mutatott Windows API függvények bizonyára más nyelveken is elérhetőek.

Amennyiben programunk valamilyen hibát észlel, esetleg elvégez valamilyen fontos műveletet, célszerű mindezt egy naplófájlban rögzíteni. Ezzel a rendszergazdák könnyedén visszatekinthetik, hogy mi is történt a futás során, esetleg valamilyen ütemezett feladat megfelelően végezte-e el a műveleteket. Létrehozhatunk egy saját naplófájlt, melybe minden fontos eseményt feljegyzünk, célszerű azonban mindent egy központi helyen tárolni, ahol a rendszergazdák könnyedén, egy helyen láthatják valamennyi alkalmazás naplóját. Ehhez nyújt segítséget a Windows NT/2000/XP Eseménynapló szolgáltatása.

* Esemény típusok

A Windows öt különböző eseménytípust definiál, melyekhez kötelezően, minden típushoz tartozó információkat tárolunk, valamint kiegészíthetjük típus specifikus információkkal. Minden jelentett esemény egy bizonyos típusú lehet. Az Eseménynapló program ezt a típusjelzőt használja arra, hogy a megfelelő ikont jelenítse meg a listában.

1. Információ: Ezek olyan események, melyek egy fontosabb művelet sikeres befejezését jelzik. A legtöbb szerver szolgáltatás például naplózza, ha sikeresen elindultak, vagy leálltak, de például egy általánosan használt program, amit a felhasználó napjában többször is elindít, nem célszerű, hogy minden indítását és leállását naplózza, mivel ez jelentéktelen művelet, fölöslegesen teleszemeteli az eseménynaplót.
2. Figyelmeztetés: Ezek olyan programhibákat, problémákat jelölnek, amelyek jelenleg nem okoznak feloldhatatlan hibát, azonban a későbbiekben komoly gondok adódhatnak, ha nem néznek utána. Például ilyen figyelmeztetés típusú bejegyzést készít a Windows, ha vészesen fogy a szabad terület valamelyik merevlemezünkön.
3. Hiba: Ez már olyan hiba, ami fellépésekor gondot okozott, esetleg adatvesztés is történt. Például ha egy fontos fájlt nem sikerült menteni, vagy egy szolgáltatás nem tudta megfelelően inicializálni magát.
4. Sikeres bejelentkezés: ez az esemény akkor történik, amikor valamilyen auditálási folyamat (például bejelentkezés a rendszerbe) sikeresen zárult.
5. Sikertelen bejelentkezés: amennyiben a fenti folyamat hibával ért véget, ezt az esemény típust kell használnunk.

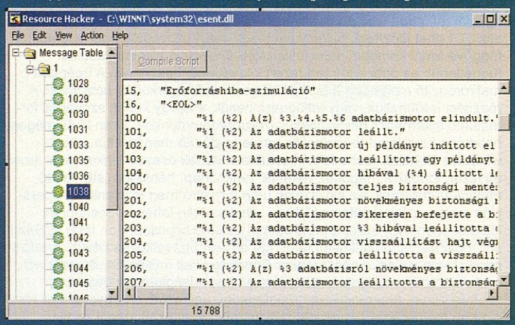


* Naplófájlok

A Windows többféle naplófájlt tart fenn az alkalmazások számára, hogy az egyes napló bejegyzések valamilyen csoportosításban legyenek megtekinthetőek. Az Application (Alkalmazás) napló minden általános üzenetet, hibát tartalmaz, ezt használja a legtöbb program. A System (Rendszer) ---- naplót főként a szolgáltatások, eszközmeghajtók használják. A Security (Biztonság) naplót a rendszer használja, nekünk csak olvasási hozzáférésünk van hozzá. Lehetőség van egyéni napló létrehozására is, ezzel majd később foglalkozunk.

* Esemény források

Minden bejegyzéshez meg kell adnunk, hogy honnan származik. Ez tulajdonképpen alkalmazásunk neve, melynek segítségével még áttekinthetőbbé válik a napló, hiszen nem csak az üzeneteket látjuk, hanem azt is, hogy melyik alkalmazás generálta azt. Minden ilyen esemény forrást regisztrálnunk kell. Ehhez először is egy Registry alkulcsot kell létrehoznunk a HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog\XXX kulcsba. Ahol XXX lehet Application, vagy System, attól függően, hogy melyik naplófájlhoz szeretnénk a forrást megadni.
Az így létrehozott kulcsban lehet megadni, hogy az egyes bejegyzéseket milyen kategóriákba soroljuk (a későbbi szűrést segítve ezzel), valamint hogy melyik fájlban találhatóak az üzenetek. Ezekről az üzenet fájlokról elöljáróban annyit, hogy minden nyelvhez (pl. 1038 = magyar) tartalmazzák az üzeneteket és minden üzenetnek van egy sorszáma, egy kódja, amit a későbbiek során használunk fel.



Az esemény forrásokkal bővebben, valamint az üzenet fájlokkal a következő részben ismerkedünk meg közelebbről.

Geiger Tamás
info@gsc.hu

VERSENYEK

Évente megrendezésre kerülö programozással kapcsolatos versenyek

Nemzetközi 24 órás Programozóverseny

A Budapesti Műszaki és Gazdaságtudományi Egyetemen szervezésében évente. tavasszal megrendezett verseny elsődleges célja, hogy hazai és külföldi informatikus hallgatóknak és már végzett, az iparban dolgozó profiknak lehetőséget nyújtson képességeik és önmaguk határainak megismerésére extrém körülmények között.

A magas összdíjazású verseny háromfős csapatok között zajlik, melyet egy Internetes előválogatás előz meg, és pontosan 24 órán át tart. A számos részfeladat egy probléma köré épül, melynek megoldása a versenyzők részéről széles körű informatikai ismereteket igényel, ideértve a programtervezést, kódolást, hálózati ismereteket, valamint a mesterséges intelligenciát is.

További információ: Benedek Balázs – főszervező: http://www.challenge24.org

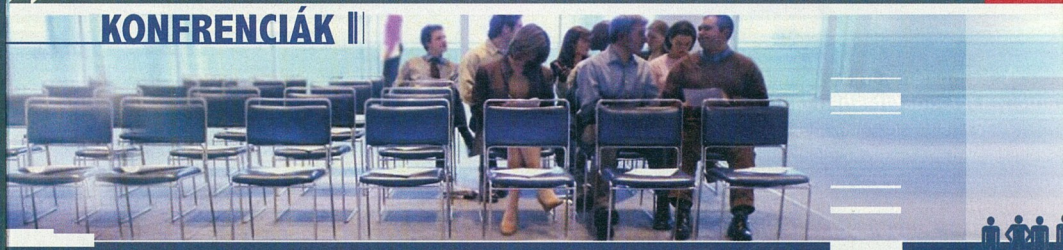
Abakusz - szoftverfejleszt és tehetségkutató verseny

A verseny célkitűzése a probléma- és feladat-megoldási, valamint dokumentálási képesség felmérése a szoftverfejlesztés és a szoftvertesztelés területén.

A miden év tavaszán megrendezett verseny alapkoncepciója a nyílt dokumentáció és forráskód. Célunk, hogy a beadott munkák a tanulás szolgálatában álljanak, és az elbírálás igazságos legyen. A helyezett versenyző számára komoly reklám lehet munkájának közzététele. A copyright természetesen a versenyzőé marad.

További információ a Gábor Dénes Főiskola honlapján található: http://www.gdf.hu/abakusz/

KONFERENCIÁK



XIV. Országos Térinformatikai Konferencia

A térinformatikai alkalmazások egyik legnagyobb szabású hazai rendezvényére kerül sor 2004. szeptember 23-24-én, tizennegyedik alkalommal Szolnokon.

A konferencia célja az, hogy néhány kiemelt témakör vonatkozásában, elsősorban a közigazgatásra fókuszálva, esettanulmányokon keresztül a térinformatikai alkalmazások, és azok gyakorlati tapasztalatai kerüljenek bemutatásra, nem megfeledkezve a legújabb ismeretek átadásáról sem. Az előadások mellett munkaműhely próbálja meg a résztvevők aktív bevonásával összegezni a tapasztalatokat, közérdeklődésre számot tartó témakörökben.

További információk a http://www.otk.hu/home.asp

INFO Alapítványok konferenciái

Az INFO ÉRA Alapítvány és az INF.O. Plus Alapítvány évente két alkalommal, tavasszal és ősszel hívják meg az érdeklődőket informatikai-oktatói rendezvényeik részvételére, többek között a következő témakörökben: ***Logo, adatbázisok a Web-en, hálózatok az iskolában, az informatika pszichológiai vetületei, információs társadalom, Web-alkalmazások, új technológiák, flash, digitális fényképezés stb.***

További információk a http://www.infoera.hu/ webcímen érhetők el.

MORFEUS

A Magyar Operációs Rendszer Fejlesztok Szövetsége és Érdekvédelmi Képviselete (a továbbiakban "MORFEUS", "szövetség") elsődleges célja olyan bel- és külföldi magyar szoftverfejlesztőket szakmai szervezetbe tömöríteni, akik operációs rendszerek-, illetve azzal szomszédos szoftverek fejlesztésével, oktatásával, új technológiák kutatásával foglalkoznak. A szövetség támogat néhány ismeretterjesztő folyóiratot is. Mindezeken felül érdekvédelmi képviseletként a MORFEUS olyan szövetség kíván lenni, amely szakmai,

Minden érdeklődőt tisztelettel várnak a http://www.morfeus.hu/ honlapon.

A MAGYAR FELSŐOKTATÁS
MODERN SZOLGÁLTATÓ KÖZPONTJA

Kempelen Farkas
Hallgatói Információs Központ (HIK)

A Kempelen Farkas Hallgatói Információs Központ (HIK) olyan, Magyarországon ***egyedülálló intézmény, amely a felsőoktatásban tanulóknak, oktatóknak, kutatóknak nyújt európai színvonalú szolgáltatásokat.***

A "21. századi tanulóhely" olvasó-, és oktatótermeiben 350 multimédiás számítógép áll rendelkezésre, melyek segítségével elérhetők az elektronikus adatbázisok (EISZ) és az Internet. Továbbá könyvtár, könyvesbolt és Felsőoktatási Információs Pont (FIP) várja a látogatókat. A főváros belvárosban (VIII. ker. Reviczky u. 6.) található központ ideális tanulóhelyet teremt kellemes, ***nyugodt környezetével és ingyenes, vagy kedvezményes szolgáltatásaival (nyomtatás, fénymásolás, kulturális programok, stb.***).

Minden tanulni vágyót szeretettel várnak!

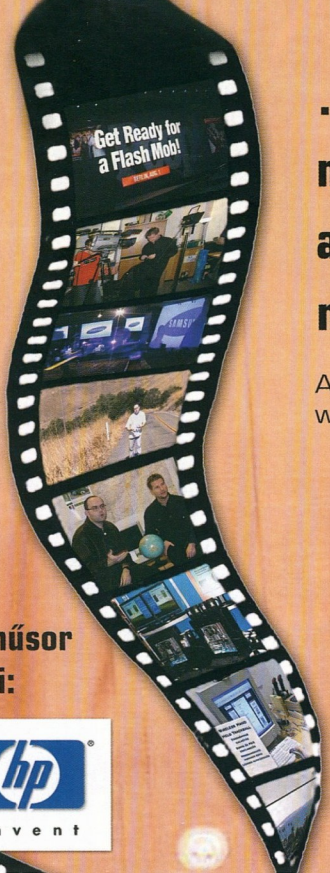
További információk a http://www.hik.hu weboldalon találhatók.

Nyakkendő nélkül a számítástechnikáról...

PC STUDIO

...30 perces TV műsor, minden hónapban a PC World CD/DVD mellékletén!

Az összes adás letölthető:
www.pcworld.hu/pcstudio



Most érdemes DVD-s PC Worldre váltania!

2 hónap múlva már csak a DVD-s változaton nézheti meg a PC Studio adásait!

A PC Studio műsor támogatói:

Logitech

hp invent