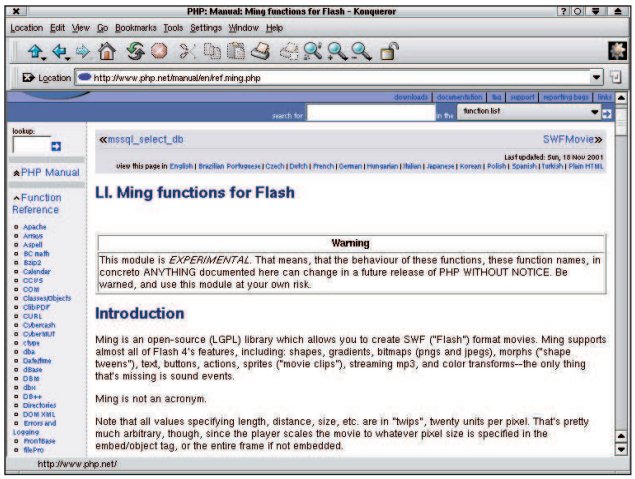
A PHP és a MING

Hogyan készítsünk weblapunkra röptében Flash-mozikat?

Napjainkban a Flash-féle pörgő-forgó csodát nehéz kikerülni a Világhálón. Nem is kell, hiszen a Flash-lejátszás az összes újabb linuxos böngészőben lehetséges. Kedvenc operációs rendszerünkön manapság az ilyen mozik létrehozása sem elérhetetlen cél. A ***MING*** könyvtár segítségével és PHP-támogatással weblapjainkat elláthatjuk röptében előállított animációkkal. Cikkünkben a MING Linuxra telepítéséről, valamint a PHP-vel történő összeházasításáról lesz szó. Emellett a cikk kínálta lehetőségekhez mérten igyekszem a MING működését is bemutatni.

A MING beszerzése és telepítése

A PHP MING kiegészítésének telepítését Debian Woody rendszeren a PHP 4.0.6-os változatához mutatom be. Kis módosításokkal természetesen más GNU/Linux-változatokon is üzembe helyezhető. Tekintettel arra, hogy a támogatás csak a 4.0.5-ös változattól került a PHP-ba, érdemes a gépünkön egy friss PHP-val próbálkozni.



1. kép A MING-függvények részletes leírása a PHP-kézikönyvben

A MING hivatalos weblapját megnyitva hamar szembesülhetünk vele, hogy nem vagyunk magunkra hagyatva. Innen a legfrissebb PHP-változatokhoz azonnal letölthetjük az előre lefordított ***php_ming.so*** modult. Jó esetben akár lusták is lehetünk, és a MING-kiegészítést fordítás nélkül beizzíthatjuk, csupán ezt a fájlt szükséges a PHP-kiterjesztéseket tartalmazó könyvtárba másolnunk. Ennek helyét könnyen megtudhatjuk, ha a parancssorban kiadjuk a `php-config --extension-dir` parancsot. A másolás mellett még a ***php.ini*** fájlba is bele kell nyúlnunk, és a következő bejegyzést kell beillesztenünk:

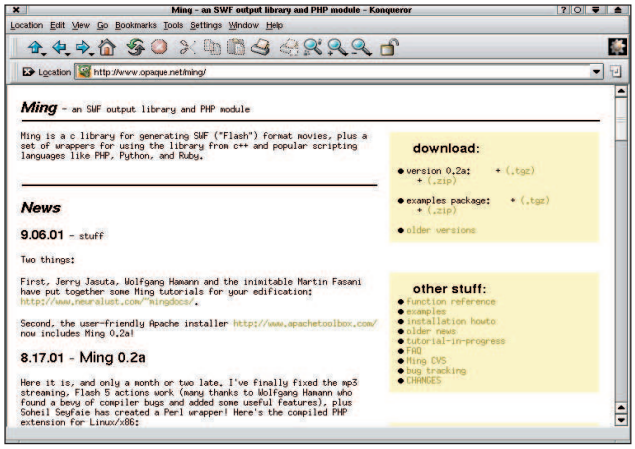
```
extension=php_ming.so
```

Amennyiben ezzel megvagyunk és minden egyezik, máris rendelkezünk a Flash-mozik létrehozásához szükséges eszköztárral. Ne feledkezzünk meg webkiszolgálónk újraindításáról, amennyiben az a beállításváltozások érvényre juttatásához szükséges.
Előfordulhat, hogy a folyamat nem ilyen egyszerűen zajlik le, ekkor sem kell kétségbe esni, a ***php_ming.so*** kiegészítést mi magunk is könnyen létrehozhatjuk. Ehhez először be kell szereznünk a MING forráskódját, majd le kell fordítanunk és telepítenünk (ehhez a parancsokat a MING forráskönyvtárából adjuk ki):

```
make
make install
```

Ezáltal a ***/usr/lib*** alá létrejön a ***limbing.so*** állomány, és egy ***ming.h*** is megfelelő helyére kerül a ***/usr/include*** könyvtárban. Ezáltal megteremtettük a feltételét, hogy a MING-támogatást a jelenlegi PHP-rendszerünkbe építsük. A további szükséges lépéseket már a PHP-forráskönyvtárából kell elvégeznünk:

```
./buildconf
./configure --with-ming <egyØb kapcsol k>
make
make install
```



2. kép A MING oldala ➲ http://www.opaque.net/ming/

A szükséges könyvtárat tehát létrehoztuk, és a helyére is került. A ***php.ini***-nek a fenti bejegyzéssel történő kiegészítése természetesen ekkor is szükséges.

Az ismerkedés

A nagy fejesugrás előtt nem árt néhány dolgot tisztázni: a pontosság és a jó nagyíthatóság érdekében bevezették a ***twip*** mértékegységet. Hogy értsük, mit is takar ez: húsz twip tesz ki egy képpontot. A mozi méretei, azon belül is minden elemnek a mérete, a távolságok mind ebben az egységben értelmezendők. Alapesetben tehát egy 200×200 képpontmérettel rendelkező mozihoz 4000×4000 twip méretű valódi munkafelület tartozik. A másik fontos tudnivaló, hogy a MING jelen pillanatban csak

az FDB-típusú betűkészletek megjelenítésére alkalmas. Ilyenek begyűjtésére a MING-forrás ***util*** alkönyvtárában található `makefdb` használható, először ezt sem árt lefordítanunk. Tekintsük át nagy léptékekben, hogyan is épül fel a MING-birodalom! A legtöbb PHP-kiegészítéssel ellentétben itt nem ömlesztett függvénykönyvtárat kapunk a nyakunkba, hanem 13 osztályt. Ezek mindegyike egy témát ölel fel, és a hozzá tartozó eljárásokat, függvényeket, tulajdonságokat hordozza magában. Lássuk, miből fogunk csipegetni!

`SWFShape()`	Ezzel hozhatjuk létre és rajzolhatjuk meg a különböző, a moziban megjelenő formákat.
`SWFBitmap()`	A moziba bevihető objektumokat JPEG-képekből hozhatjuk létre.
`SWFText()`	A szöveges elemek létrehozására való osztály.
`SWFTextField()`	Szöveges űrlapelemek létrehozásához.
`SWFSprite()`	Önálló animációs almozik hozhatók létre vele, amelyek saját időskálával rendelkeznek.
`SWFButton()`	Nyomógombok létrehozására szolgál.
`SWFFont()`	Különféle betűtípusokat tölthetünk be vele, valamint a szövegek megjelenítéséhez szükséges.
`SWFGradient()`	Színátmenetek létrehozására, továbbá a formák kifestésénél használható.
`SWFFill()`	Már meglévő kifestőobjektumok forgatására, mozgatására és átméretezésére szolgál.
`SWFDisplayItem()`	A moziban létrehozott, behúzott objektumokat itt tudjuk pörgetni, forgatni és nagyítgatni.
`SWFMorph()`	Alakjukat változtató látványelemek létrehozását teszi lehetővé.
`SWFAction()`	A Flash saját nyelvén írhatunk ActionScripteket.
`SWFMovie()`	A mozi maga: elemeket adhatunk hozzá, menthetjük vagy a kimenetre küldhetjük a tartalmát.

Az `SWFMovie()` osztályt mindig használni fogjuk, mert mind a mozi születésekor, mind a véglegesítésekor jelen van. Egy moziobjektumot a következő módon hozunk létre:

```
$mozi = new SWFMovie();
```

Innentől létezik is $mozi néven az objektumunk, ebbe szórjuk bele a mozgatnivaló elemeket. Ha létrehoztuk, nem árt néhány vele kapcsolatos dolgot beállítani:

```
$mozi->SetRate(20);
$mozi->SetDimension(4000,4000);
$mozi->SetBackground(0xff, 0xaa, 0x66);
```

A `SetRate()` által tudjuk megadni, hogy egy másodpercben hány képkockát játszunk le, ez lesz a teljes mozira érvényes beállítás. Megjegyzendő, hogy csak amolyan kívánatos értékről van szó, hiszen egy leterhelt gépen a képkockák megrajzolása a rendelkezésre álló időnél többet vehet igénybe. Ilyenkor egyszerű lassulásról van szó: a lejátszó nem hagy ki kockákat, csupán lassabban játssza le őket. Más eset áll fenn akkor, ha folyamatos MP3 zenei aláfestés is tartozik a mozinkhoz, mert

1. lista A gorillamozi.php – immár teljes pompájában

```
<?php

$mozi = new SWFMovie();

$mozi->SetRate(20.0);
$mozi->SetDimension(4000,4000);
$mozi->SetBackground(0xff, 0xaa, 0x66);

// A kimenetre k ldØs elıtt tudatnunk kell
// a b ngØszıvel az adathalmaz MIME-t pusÆt.

header('Content-type:
       ➥application/x-shockwave-flash');

$mozi->Output();

?>
```

2. lista A gorilla.html – az első mozinkat beágyazó HTML-oldal

```
<html>
<head>
    <title>GorillÆk - rendezıi
      ➥vÆltozat</title>
</head>
<body bgcolor=#ffffff>

    <embed src=gorillamozi.php width=200
      ➥height=200></embed>

</body>
</html>
```

ilyenkor a lejátszónak biztosítania kell, hogy a hanglejátszás lehetőleg folyamatos legyen. Ezt a gondot képkockakihagyással küszöböli ki.

A befoglaló méreteket, azaz megjelenő munkaterületünk méretét a `SetDimension()` által adhatjuk meg. Mint korábban már említettem, itt nem képpontokban, hanem twipekben kell gondolkodnunk, azaz a fenti példa egy 200x200 képpont méretű Flash-objektumot hoz létre.

A méretek közül először a szélességet, másodjára a magasságot kell megadni. A mozinak kell háttérszín is. Ennek beállításához használatos a `SetBackground()`. A háttérszínt az RGB- (vörös, zöld, kék) összetevők keverésével tudjuk létrehozni. A színeket 3x8 biten képezhetjük le, vagyis az egyes értékek értéktartománya 0-tól 255-ig terjed, és csakis egész számokban adhatók meg. A példában éltem a PHP nyelv adta lehetőséggel, és az értékeket tizenhatos számrendszerben ábrázoltam (hiába no, én már csak hexadecimálisokban látom a színeket).

Örvendezzünk, ugyanis elkészítettük életünk első egész estés animációs filmjét! A címe „Gorillák a narancsos ködben" lehetne, tekintve az események letisztult egyszerűségét. Egy apróság hiányzik még: mozinkat láthatóvá is kellene tenni a közönség (legalábbis a böngészőnk) számára. Ehhez az

3. lista A haromszog.php már valami, de még nem mozog…

```
<?php

// A befoglal  mozi.

$mozi = new SWFMovie();
$mozi->SetDimension(6000,6000);
$mozi->SetBackground(0xff, 0xff, 0xff);

// a hÆromsz g megrajzolÆsa-

$valaki = new SWFShape();
$valaki->setLine(5, 0xce, 0xce, 0xce);
$valaki->setRightFill($valaki->addFill(0xe0,
      ➥0xe4, 0xec, 50));
$valaki->movePenTo(0,1600);
$valaki->drawLineTo(-1000,-400);
$valaki->drawLineTo(1000,-400);
$valaki->drawLineTo(0,1600);

// tegy k a moziba, Øs k ldj k a helyØre

$haromszog = $mozi->add($valaki);
$haromszog->move(3000,3000);
$mozi->nextFrame();

// kØsz van, mehet a vilÆg elØ...

header('Content-type:
      ➥application/x-shockwave-flash');

$mozi->Output();

?>
```

`SWFMovie()` egy újabb fontos eljárását kell alkalmaznunk. Lássunk erre is példát!

```
$mozi->Output();
```

Munkánk eredményét ez a gyakorlatlanok számára így még eléggé emészthetetlen formában tálalja: krixkrax karakterek halmazaként. Mielőtt még e furcsa betűkben látni kezdenénk a jeleneteket, tegyük fogyaszthatóvá az adatokat. Ehhez több dologra is szükség lesz: először is tudatnunk kell a böngészővel, hogy a küldött adatfolyam Flash-mozit közvetít. Második lépésként pedig létre kell hoznunk egy, a művünket beágyazó HTML-oldalt.
Tapasztalat, hogy a Flash-mozik fejlesztése során böngészőnk gyorsítótárát érdemes kikapcsolni, ellenkező esetben hajlamos makacsul ragaszkodni egy korábbi állapothoz. Így pedig meglehetősen nehézkes a kódolás folyamán ellenőrizni, hogy az történik-e, amit valóban szeretnénk.

Lejátszó hiányában…

Előfordulhat, hogy jelenlegi böngészőnk nem alkalmas Flash-fájlok lejátszására, ilyenkor hamar átirányít a Macromedia letöltési oldalára. Amennyiben ez önműködően nem történne meg, magunknak kell ellátogatnunk oda (*3. kép*).

Forgatás előtt – a szereplők

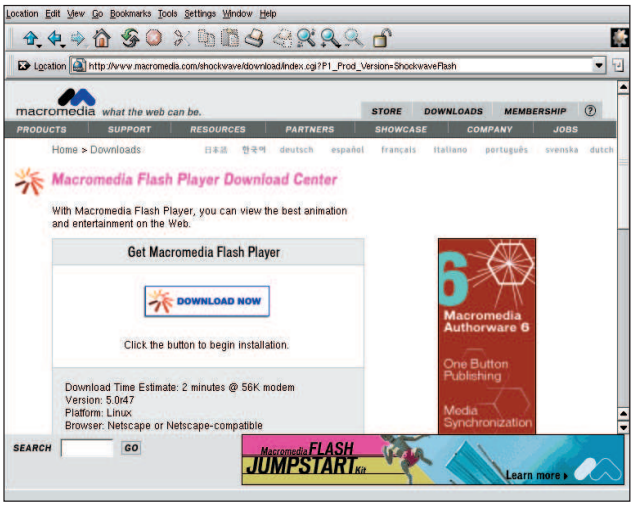
Most már feltehetően az összes szükséges eszközzel rendelkezünk a PHP-s Flash-fejlesztéshez. Beállítottuk a kiszolgálót és a böngészőnket is. Kíséreljünk meg összeállítani valami komolyabbat. Eddig csak az `SWFMovie()` osztály biztosította eszközkészlettel ügyeskedtünk. Itt az ideje megismerni egy másik, szintén elég alapvető osztályt, az `SWFShape()`-et.
A szemléltetést egy egyszerű háromszög alakjának megrajzolásával kezdem. Hozzuk létre a formát képviselő objektumot:

```
$valaki = new SWFShape();
```

Innentől $valaki néven létezik az objektum. Miután megrajzoltuk, az animációnkba számtalan példányban beilleszthetjük (így lehet fenyőfákból erdőt növeszteni). Természetesen minden ilyen egyed külön torzítható, forgatható, mozgatható. A forma megrajzolása előtt szükség lesz pár alapadat megadására, ilyen például a vonalvastagság és -szín. Tudatnunk kell azt is, ki akarjuk-e a rajzunkat festeni, és amennyiben igen, vajon mivel.

```
$valaki->setLine(5, 0xce, 0xce, 0xce);
$valaki->setRightFill($valaki->addFill(0xe0,
      ➥0xe4, 0xec, 50));
```

Az első sorral a rajzot megjelenítő vonal stílusát határozhatjuk meg. Először a vonal vastagságát, majd pedig a színét kell megszabnunk. A vastagság természetesen twipben értendő, ezért az 5-ös vastagság elég vékonynak számít. A vonal színének megadása a korábban megismerthez hasonló módon zajlik.



3. kép ➲ http://www.macromedia.com/shockwave/download

A vonalrajzolás beállítása után következik a kifestés meghatározása. Mint látható, egymásba ágyazott eljárásokról van szó. A belsővel, azaz az `SWFShape()` osztály `addFill()` függvényével különféle kifestési stílusokat hozhatunk létre. A jelenlegi egy puritán egyszínű festőstílus. Három adatot kell kötelezően megadnunk, amelyeket akár egy negyedikkel is kiegészíthetünk. A példára tekintve az első három talán kézenfekvő is, ezek a szokásos színösszetevők. A negyedik megadható adat pedig egy 0-tól 100-ig terjedő, áttetszőséget meghatározó érték. Ennek a `$valaki->addFill()` kifejezésnek a visszatérő értékét (egy azonosítót) kapja meg a `$valaki->setRightFill()`.
A `setRightFill()` parancsnak létezik egy társa, a

4. lista A mar_valami.php – mozgással ellátott mozi

```
<?php

// vØletlenszÆmgyÆrt  f ggvØny behœzÆsa

include 'random.inc';

// beÆll tÆsok egy helyen

define('NUM_TRIANGLES',20);
define('NUM_FRAMES',100);
define('FRAME_RATE',15);

// kezdı lØpØsek: mozi lØtrehozÆsa

$mozi = new SWFMovie();
$mozi->SetRate(FRAME_RATE);
$mozi->SetDimension(6000,6000);
$mozi->SetBackground(0xff, 0xff, 0xff);

// a mozi sorÆn rengetegszer felhasznÆlt
// hÆromsz g alakjÆnak megrajzolÆsa

$valaki = new SWFShape();
$valaki->setLine(5, 0xce, 0xce, 0xce);
$valaki->setRightFill($valaki->addFill(0xe0,
➥0xe4, 0xec, 50));
$valaki->movePenTo(0,1600);
$valaki->drawLineTo(-1000,-400);
$valaki->drawLineTo(1000,-400);
$valaki->drawLineTo(0,1600);

// A k vÆnt mennyisØgß hÆromsz g elhelyezØse,
// vØletlenszerß torz tÆsa, szØtsz rÆsa.
// Mindegyikhez k l nb zı,  sszevissza
// lØtrehozott forgÆsi sebessØg adÆsa.

for ($i=0; $i<NUM_TRIANGLES; $i++) {
    $hszog[$i] = $mozi->add($valaki);
    $hszog[$i]->move(2000+randomint(2000),
            ➥2000+randomint(2000));
    $hszog[$i]->scale(randomint(15)/10,
            ➥randomint(30)/10);
    $hszog[$i]->rotate(randomint(360));
    $hszogforg[$i] = randomint(15)-7.5;
}

// a k vÆnt mennyisØgß kØpkocka legyÆrtÆsa
// szØp sorban. Itt mÆr csak forgatni kell. :)

for ($j=0; $j<NUM_FRAMES; $j++) {
    for ($i=0; $i<NUM_TRIANGLES; $i++) {
      $hszog[$i]->rotate($hszogforg[$i]);
    }
    $mozi->nextFrame();
}

// kØsz van, mehet a vilÆg elØ...

header('Content-type:
            ➥application/x-shockwave-flash');

$mozi->Output();

?>
```

`setLeftFill()`. Ha formánk pontjait sorrendben úgy adjuk meg, hogy a körbejárási sorrendjük az óramutató járásával megegyező irányú, akkor van szükség az elsőre. Fordított irányban haladva a „befelé” balra esik. Érdemes erre figyelmet fordítani, mert a lejátszó esetleg bedobhatja miatta a törülközőt. Érdekes adat, hogy ezeket jelenleg valamiért az `SWFMorph()` osztályon belül felcserélve kell használnunk. Apró következetlenség, majd „kinövi” a program. No igen, a PHP/MING-leírás minden egyes oldalon kihangsúlyozza, hogy ez a modul igencsak kísérleti állapotban található, az egyes elemek bármikor gyökeresen megváltozhatnak. Így jelenleg senki sem garantálja, hogy a mostani MING-objektumaink a következő kiadással is ugyanúgy fognak működni, tehát bánjunk velük óvatosan. Innentől kezdődik a forma megrajzolása, amihez vonalakat és íveket kell sorra megadnunk. Emellett rajzeszközünket vonal rajzolása nélkül is arrébb tudjuk pakolni, amire mindjárt az elején szükségünk is lesz, hiszen a tárgyunkat nem a 0,0 pontból kezdjük rajzolni. Irány a kiindulópont!

```
$valaki->movePenTo(0,1600);
```

Ezzel az eljárással „ceruzánkat” vonal rajzolása nélkül mozgatni tudjuk. A koordináták a szokásos módon X,Y sorrendben adandók meg. Vízszintes irányban egyértelmű a helyzet, hiszen ritka eset, ha valahol nem balról jobbra nő a koordinátaérték; függőleges irányban pedig számításba kell vennünk a Besenyő Pista bácsi-féle biorobotelmélet(et): „Egyet kell kérdezni: hogy mekkora, és hogy leerű-fee vagy feerű le?” Nos, kedves Boborján, a második. Tehát Y irányban a koordinátaérték lefelé növekszik, ami pont a körbejárási irány meghatározásánál a legfontosabb (csak megemlítem, hogy a méreteket itt is twipsben kell érteni). A rendszer legalább ebben végig következetes. Lehetőségünk nyílik közvetlen vagy viszonylagos helyzetmegadásra is. Példánkban egész idő alatt a közvetlen módszert választottam, amire az eljárások nevének végén található „To” szócska utal. Viszonylagos elmozdulást egy `$valaki->movePen()`-nel lehetett volna megadni. Innentől kezdve a háromszöget az óramutató járásával megegyező irányban rajzolom körbe:

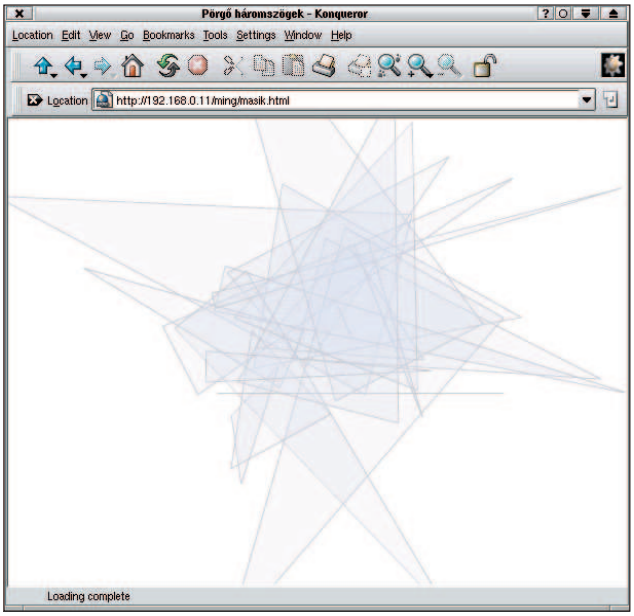
```
$valaki->drawLineTo(-1000,-400);
$valaki->drawLineTo(1000,-400);
$valaki->drawLineTo(0,1600);
```

Tárgyunk megrajzolásával ezennel végeztünk is. Háromszögünket helyezzük a munkatérbe, hogy láthatóvá váljon a kotyvasztásunk végeredménye. A befoglaló HTML-oldal elkészítését a nyájas olvasó/alkotó képzelőerejére bízom. A programlistában három ismeretlen sor található.

5. lista A random.inc a véletlenszám-előállításhoz

```
<?php
function randomint($max) {
  static $startseed = 0 ;
  if (!$startseed) {
    $startseed =
➥(double)microtime()*getrandmax() ;
    srand($startseed) ;
  }
  return (rand()%$max)+1 ;
}

?>>
```



4. kép Ez lesz a vége…

Az első helyezi tárgyunkat a mozi vásznára. Erre ezután a `$haromszog`-objektumon keresztül hivatkozhatunk. Érdemes tisztázni, hogy a `$valaki` csak a formát képviseli. Ebből számtalan egyedet helyezhetünk ki a mozivászonra, amelyeknek mind saját objektumuk lesz, hogy külön-külön lehessen őket módosítgatni. Egy ilyen egyed már az `SWFDisplayItem()` osztályra tartozik. A következő, `$haromszog->move(3000,3000)` paranccsal kerül a háromszög a munkaterület közepére. A `move` az első eljárás, amellyel az `SWFDisplayItem()` parancsai közül megismerkedhetünk. A példa kedvéért szokásomtól eltérve viszonylagos elmozdulást adtam meg. Ennek a `move()` utasításnak tehát létezik egy `moveTo()` testvére is. A formát természetesen eleve olyan módon is rajzolhattuk volna, hogy a közepe a 3000,3000 pontban legyen, de a saját 0,0 pontjának kitüntetett szerepe van: ekörül forog ugyanis, ha egy kicsit „megtekergetjük”.
A `$mozi->nextFrame()` kiadása akkor szükséges, amikor a képkockát befejeztük, azaz minden a helyén van. Mivel egyetlen kockánk készült el, véglegesíthetjük. Igaz, nem lesz több képkocka, „de úgy szép, ha kerek”, és így biztos nem adódik vele gond.

Csapó!

Az animációt kockáról kockára lépegetve kell megterveznünk. Amennyiben valami a mozivászonra került, azt ugyanabban az állapotában a következő kocka is tartalmazza. Csupán új elem szerepeltetésekor kerül sor újra az `SWFMovie()->add()` eljárás alkalmazására. Amennyiben valamit ki kell venni, az `SWFMovie()->remove(a tÆrgy azonos t ja)` parancsot kell kiadnunk.
Írásomat egy teljes animáció bemutatásával zárom, amely a *3. listában* látható példa továbbfejlesztésével készült. Forgatást is alkalmazok benne, amit az `SWFDisplayItem()->rotate()` parancsával tudok megtenni. A forgatási szög fokban értendő, és erre a feladatra a `rotateTo()` is használható. A pozitív forgatási irány az óramutató járásának megfelelő. A programban sok a véletlenszerű elem. Egyrészt a kirakott húsz háromszög sem mind a 3000,3000 ponton csücsül, hanem véletlenszerűen vannak szétszórva. A háromszögek objektumai a `$hszog` tömbbe kerülnek. Egy másik tömbbe helyezem a háromszögek forgási sebességét, amelynek neve `$hszogforg`. Szintén véletlenszerű adatokról van szó, értékük pozitív és negatív egyaránt lehet. Emellett a tárgyak kiinduló elfordulását is megadom, hogy induláskor se merevedjenek vigyázzállásba. Hab a tortán, hogy mindet torzítottam is, amire az `SWFDisplayItem()->scale()` eljárása ad lehetőséget. X és Y irányban a nagyítási arányt külön kell megadni.
A `scale()` használatával a jelenlegi mérethez képest történő nagyítás arányáról van szó. Amennyiben az eredetihez akarjuk viszonyítani, a `scaleTo()` is rendelkezésre áll.
A következő ciklus hivatott a százképkockás mozit legyártani. Minden kockán belül a háromszögeket egyesével meg kell forgatni a hozzá tartozó forgási sebesség szerint. Ha mind a húszat megmozgattuk, jöhet a következő kocka. Amint az összes kocka elkészült, jöhet a film bemutatása. A programot úgy írtam meg, hogy könnyedén lehessen játszadozni a lejátszási sebességgel, valamint a háromszögek számával és a mozi hosszával. A MING még számos haszonnal bír, ezeknek sajnos most nem jutott hely, de a PHP-kézikönyv MING-re vonatkozó részének értő olvasgatása már nem fog gondot okozni.
Kellemes ünnepeket, és jó MING-elést!

Heilig (Cece) Szabolcs (cece@mail.uti.hu) Veszprémben él, huszonhat éves fejjel már hatszoros nagybácsi. Több cégnek dolgozik PHP-programozóként, de PHP-távoktatást is végez. Linuxot először 1994-ben látott, kezdő perles szárnypróbálgatásai után 1997-ben szeretett bele a PHP-be. Szabadidejében hajlamos kerékpárra pattanni, vagy baráti társaságban szerepjátékokkal foglalatoskodni.

Kapcsolódó címek

A PHP-kézikönyv MING-része ➲ http://www.php.net/ming

A MING hivatalos oldala ➲ http://www.opaque.net/ming

A Flash formátumának birtoklója, a Macromedia oldala ➲ http://www.macromedia.com

Minden, ami a Flash formátumáról tudható ➲ http://openswf.org