DIPLOMAMUNKA

Mikó Krisztián Károly

Debrecen
2011

Debreceni Egyetem – Informatikai Kar

A Flash multimédiás lehetőségei

Témavezető:
Dr. Tornai Róbert
Egyetemi adjunktus

Készítette:
Mikó Krisztián Károly
Programtervező matematikus

Debrecen
2011

Tartalomjegyzék

1. Bevezetés

A számítógépek nagy hatású fejlődése során már nem volt elegendő a karakteres megjelenítés és megnőtt az igény arra, hogy valami másra is használják őket. Kifejlődtek a grafikus felületek, melynek hatására a multimédia kialakulása is elindult.

A gyors elterjedésben fontos szerepet játszott az üzleti élet és a reklámok közötti versengés, ahol a legfőbb célnak a figyelemfelkeltést tűzték ki. Az internet térhódításával mindez tovább fokozódott, hiszen ma már szinte elengedhetetlen az, hogy szép, igényes és átlátható megjelenítést mutasson az oldalunk. Ugyanakkor ma már a technika lehetővé teszi, hogy az átlag felhasználó is képes legyen ennek elérésére.

Az Adobe nem túloz, amikor azt állítja, hogy a Flash CS5 Professional „a digitális, webes és mobil eszközökre szánt hatásos és interaktív tartalmak legnagyobb tudású fejlesztőeszköze” [1]. CS5 legnagyobb újdonsága a Photoshop- és Illustrator-állományok beolvasásának képessége. A hangok, az animációk, a grafikák és az interaktivitás segítségével a Flash képes lenyűgözni, oktatni, szórakoztatni és általános információkkal ellátni bennünket. Több mint félmilliárd ember rendelkezik az ingyenes Flash lejátszóval, amely a Flash filmek megtekintéséhez szükséges. Az internethez csatlakozó számítógépek több mint 97%-án használják a Flash lejátszót külön programként vagy böngészőbe beépített kiegészítőként. Számos számítógépes játék és rajzfilmsorozat ezzel a technikával készül. A Flash tehát egy interaktív, weboldalak elkészítésére alkalmas eszköz, amely ma már ténylegesen a világ vezető internetes szerzői rendszerévé nőtte ki magát.

Diplomamunkám célja a Flash technológia, Flash animáció készítés bemutatása CS5 környezetben, különös figyelmet fordítva a multimédiás lehetőségekre. Bemutatom magát a Flash-t, főleg az újdonságokat a korábbi verziókhoz képest, valamint kitérek a Photoshop, Illustrator és a Flash Builder jelentőségeire. Ismertetem a Flash saját script nyelvét, az ActionScript-et is, azon belül is a legújabb, az ActionScript 3.0-t (AS3). Célom, hogy megmutassam, mennyire könnyedén használhatjuk az új fejlesztéseket Flash környezetben. Pár kattintásból meg tudjuk valósítani mindazt, amit eddig fáradságos munkával értünk volna el. A függelékben végül egy ActionScript forráskódot szemléltetek.

2. A Flash

A Flash elterjedésének kezdete 1996-ra tehető, amikor egy kis cég, a FutureSplash fejlesztői olyan webes multimédiaalkalmazást mutattak be, amellyel könnyedén lehetett nagyítható - kicsinyíthető vektorgrafikát beépíteni, valamint interaktív nyomógombokat készíteni a honlapokba. A technológiát a Macromedia cég felvásárolta és továbbfejlesztette. Később pedig az Adobe (1. ábra) megszerezte a Macromedia céget, és vitathatatlan, hogy mindezt a Flash miatt tette.

1. ábra - Az Adobe cég logója
Forrás: http://lva2010.inria.fr/adobe-logo.png

2.1 A Flash technológia

A szokásos pixelgrafikus képek minden egyes képpont adatait külön tárolják, míg a vektorgrafikus megoldás csak annyit mond a megjelenítő programnak, hogy rajzolj ide egy kört ekkora átmérővel és ilyen színnel. Ezáltal a körök, téglalapok és más síkidomok tárolásához sokkal kevesebb adat kell, ami gyorsítja a letöltést. És ebben rejlik a technológia zsenialitása. A fájlok olyan kompakt módon kerülnek tömörítésre, hogy lassabb Internet-kapcsolat esetén is gyorsan és nehézség nélkül megtekinthetők legyenek. Az animációk interaktív vezérlőelemekkel és akciókkal egészíthetők ki anélkül, hogy egyetlen sor szkriptet kellene írni.

A szoftver felhasználási területe a Web-oldalak szélén található egyszerű vektor animációktól (Banner) időközben teljes képernyős alkalmazássá növekedett, a teljes egészében Flash alapú oldalakat is ide értve. A Flash-ben megjelent a morfózis (Shape Morphing) képessége, így nem szükséges az animációk minden egyes fázisának megrajzolása, elegendő a kulcskockákat megadni, a köztes helyzeteket a program számítja ki. További újdonság volt az MP3 kódolású folyamatos hang, a felhasználó által módosítható szövegmezők elhelyezése a filmeken, és a továbbfejlesztett Flash műveletek, amelyekkel bonyolult játékokat, űrlapokat és kérdőíveket lehet készíteni. A Flash mozik programozását szolgálja az ActionScript. Az Adobe közzétette a Flash fájlok (.SWF) formátumát és ingyenesen letölthetővé tette a böngészőkbe beépülő lejátszó programot (plug-in), ezzel nagyot lendített a Flash elterjedésén. A vektorgrafikus képek minőségromlás nélkül, tetszés

szerint nagyíthatók. A raszteres (bitmap) képek adott felbontással rendelkeznek (2. ábra). A megjelenítés függ a felbontástól. A felbontást inchenkénti (1 inch = 2,54 cm) képpontok számával határozzuk meg. Amikor egy ilyen képet felnagyítunk, a megjelenítő program a hiányzó részeket adatokkal pótolja, ugyanis nincs kellő mennyiségű információ a kép megalkotásához. Amikor kicsinyítünk, a program képpontokat távolít el, amik torzításhoz vezetnek.

2. ábra - A vektorgrafikus képek torzítás nélkül nagyíthatók
Forrás: http://edutech.elte.hu/jegyzet/flash/

A Flash-ben minden létrehozható grafika vektoros. A vektorgrafikának két előnye van: az első és legfontosabb, hogy a fájlméret kicsi marad, így gyorsabban betöltődik, és a másik, hogy a képet bármekkorára méretezhetjük, akkor sem módosul a képminőség. A vektorgrafika nagyon jó, annak minden előnyével és hátrányával. A számítógépnek keményen kell dolgoznia a kép megjelenítéséhez. Ezért, ha az animáció sok vektorgrafikát tartalmaz, lelassul.

A rasztergrafikus (bitképes grafika) fájl az összes képpont színinformációját tartalmazza. Ennek eredményeként a raszter grafikák, vagy bitképek mindig nagy fájlok. A rasztergrafikát nem lehet olyan hatékonyan méretezni, könnyen szemcsés lesz. Nagy előnye, hogy gyorsan megjelenik a képernyőn. A kép jellegén múlik, hogy vektorgrafikát vagy rasztergrafikát használunk. Ha a kép geometrikus, tiszta színábrázolással, a vektorgrafika a jó választás. Fénykép esetén viszont a bitkép a megfelelő. Könnyű eldönteni melyiket használjuk, ha ismerjük az egyes típusok szempontjait. A Flash nagy előnyeként kell említeni,

hogy gyakorlatilag böngészőfüggetlen, vagyis a különböző böngészőprogramok mindig ugyanúgy jelenítik meg a Flash animációt, vagyis nem tartalmazza a natív HTML, illetve DHTML oldalak közismert kompatibilitási gondjait. Az Adobe-nak sikerült a Flash-t úgy programoznia, hogy a Flash filmek - függetlenül attól, hol kerülnek megjelenítésre (Mac, PC; Netscape Communicator, Microsoft Explorer, Mozilla Firefox), - mindig ugyanazt láthatja a felhasználó. Ezen kívül az Adobe egyéb eszközökre - mint pl. a Palm, iPhone - törekszik fejlesztéseivel. Ma már a mobiltelefonokon is megtalálható a technológia. Rendkívül gyorsan terjed. 2009-re a Flash olyan fejlesztési környezetté nőtte ki magát, amelynek segítségével bármilyen tartalmat létre lehet hozni, akár internetes alkalmazásról vagy mobil tartalomról. Az Adobe mobil Flash technológiája alapvetően két elemből áll: az egyik a Flash Player (lejátszó) mobileszközökre szabott változata, az Adobe Flash Lite, a másik az Adobe FlashCast kliens-szerver megoldás, amely különböző látványos interaktív tartalmak megjelenítésére alkalmas.

A Flash technológia jövője mindenképpen a RIA (Rich Internet Application). Ez egy olyan webes felület, amely webalkalmazásként működik. A jövő a Flash-ről, az Adobe Flex alkalmazásról (Flash alapú fejlesztői környezet) és az ingyenesen letölthető, következő generációs RIA fejlesztő eszközről, az Adobe Integrated Runtime (AIR) alkalmazásról fog szólni. Az AIR segítségével a programozók a legkorszerűbb technológiákat felhasználva online és offline is futtatható webes alkalmazásokat fejleszthetnek.

2.2 Flash Professional Creative Suite 5

A legújabb változat a CS5 jelzést viseli. Kezelőfelülete testreszabható és sokkal átgondoltabb, mint a korábbi változatoké [2]. A CS5 előtti verziók nagy hibája volt, hogy nem tudták kezelni a videó formátumokat. Ezért ha valaki mozgóképet akart megjeleníteni, egymás után kellett illesztenie képkockánként a mozgásfázisokat. Az új CS5 már importálja a legismertebb videofájlokat, az MPEG-et, a MOV-ot és az AVI-t. Az újrahasználható videotömörítési és -feldolgozási beállítások, mint a kivágás, méretezés, színkorrekciók és a kompresszió-paraméterezés gyorsítják és egyszerűbbé teszik a videó beágyazását és felhasználását a készülő alkalmazásokban. Lehetőségünk van, hogy hasábokkal és kétirányú szövegekkel dolgozhassunk, és a más Adobe termékekben készült szövegek is sokkal pontosabb formátumban és tördelésben kerülnek át a Flash-be. Az új szövegfeldolgozó és

megjelenítő meghajtó (FlashType text engine) lehetővé teszik nagyon éles, tökéletesen elsimított élekkel rendelkező, kis betűméretnél is jól olvasható szövegrészek elhelyezését. Tartalmaz integrált fontkirajzoló beállításokat, amelynek segítségével optimalizálhatjuk szövegeinket attól függően, hogy mozgó vagy álló (statikus) szöveget használunk az animációban. Újraszerkeszthetjük alakzatainkat fogópontok segítségével. Még realisztikus kinematikus effektusok hozhatók létre a Bones eszközzel. Egyszerű felhasználói felületen állíthatók be azok a paraméterek, melyekkel összetett fizikai szimulációkat hozhatunk létre. Fejlődött a Flash Professional és a Flash Builder viszonya. A Flash Builderben készített ActionScript programok megnyithatók a Flash Professionalben is. Erről bővebben a 6. Fejezetben lesz szó. Mialatt szerkesztjük a kódot a Flash Builderben, addig a Flash Professionalben végrehajtható a tesztelés, a hibakeresés és a publikálás művelete. Nőtt a teljesítmény. Az összetett grafikus effektusokkal elkészített Flash tartalmat is úgy tudjuk futtani a legújabb Flash Player 10.1 lejátszóval, hogy amíg tart a lejátszás, szinte észrevehetetlen a teljesítménycsökkenés. Ezt a program legújabb konvertáló eljárása teszi lehetővé, amelyet külön beállíthatunk (property inspector) vagy ActionScripttel is irányíthatunk. Úgy működik, hogy a lejátszó minden felhasznált „klip szimbólumot” (movie clip symbol) dinamikusan konvertálni tud bittérképes képpé a futásidő alatt, és ezt egyfajta kiegyenlítő memóriában tárolja (cache), csökkentve ezzel a processzor terhelését. A lejátszás során eddig a processzor állandóan újra és újra, képkockáról-képkockára átszámolta a vektorgörbék és -alakzatok elhelyezését a Flash animációban. A vektoradatok ilyesfajta feldolgozása és készenléti tárolása lehetővé teszi, hogy a lejátszás során bármikor azonnal visszatekerjük a lejátszás eredeti állapotába, vagy elérjük az alakzatok és görbék eredeti, kiinduló formáját. Fejlődtek és gyorsultak az „idővonal-hatások”, amelyek csökkentik az explicit kulcskép-kockák szükségességét az animálás és módosítás során. Lehetőségünk van egy lépésben véghezvinni az egyszerű munkafolyamatok ismétlődő lépéseit. Az effektusok nem destruktívak, így azokat mindig újra módosíthatjuk, vagy akkor is eltávolíthatjuk, ha már egyszer használtuk őket. Ilyenek az átmenet (transition), transzformáció (transform), segédrácsra másolás (copy to grid), osztott másolás (distribute duplicate), homályosítás/elmosás (blur), vetett árnyék (drop shadow), tágítás (expand) vagy szétrobbantás (explode).

Az Adobe Creative Suite CS5 (3. ábra) egy olyan csomag, amely teljes megoldást jelent a web-szerkesztők, a kreatív grafikai stúdiók, videós szakemberek, web-fejlesztők és mindenki számára, aki nyomdai, internetes, mobil, multimédiás, vagy videó kiadványszerkesztéssel foglalkozik.

3. ábra - Az Adobe CS5 logója
Forrás: http://www.adobe.com/images/

„A Creative Suite nem csak a szoftverek egy dobozba csomagolását jelenti, hanem a programok teljesen összehangolt munkavégzését is.” Könnyű az alkalmazások közötti átjárás és biztonságos, mert azok teljes összhangban kezelik egymás fájljait és ugyanazt a színkezelést használják. Nem kell többé ideiglenes és átmeneti állományokkal küszködni. Az Adobe InDesign például beolvassa az Adobe Illustrator (.AI) és az Adobe Photoshop (.PSD) formátumokat is. A régebbi Macromedia Studio és web-es programjai is a csomagba beépülésre kerültek. De a fejlesztés nem áll meg a nyomtatásnál és az internetnél. Az Adobe Creative Suite segítségével mobil eszközökre is készíthetünk animációt és grafikát. Ugyanígy a csomag része a professzionális videó szerkesztő kollekció is.

Az Adobe Flash Professional CS5 egy olyan professzionális webfejlesztő és webtervező program, amely a weboldalaktól kezdve, az online alkalmazásokon át, a mobil eszközökig, segítik az online tartalom fejlesztését. A Flash-nek köszönhetően látványosabb, interaktívabb és elérhetőbb weboldalakat lehet tervezni bármely ágazat számára, legyen az a szórakoztatóipar, a közigazgatás, az oktatás vagy a fogyasztási cikkek. A Flash révén olyan vektor-animációs eszköz került a fejlesztőkhöz, felhasználókhoz, amely először tette lehetővé az akkor még statikus weben a mozgó grafika egyszerű bevezetését.

3. Flash multimédiás eszközei

3.1 A multimédia definíciója

Kezdjük azzal, hogy definiáljuk, mit értünk a multimédia alatt. A multimédia divatszó, a számítástechnika egyik gyorsan fejlődő alkalmazási területe, ami alatt szövegnek, állóképnek, hangoknak, animációknak és videofilmeknek a számítógépen történő használatát értjük. A multimédia fokozza a felhasználó élményeit, gyorsabbá teszi az információ felhasználását. Hangokkal, animációkkal és videofilmekkel korábban a szórakoztató elektronika foglalkozott, mert ezek analóg alkalmazások voltak. A szórakoztató elektronikában terjed a digitális technika, a szórakoztató elektronika és a számítástechnika közeledik egymáshoz.

Az adatátvitel is kapcsolódik a multimédiához. Az Interneten gyorsan és könnyen lehet szöveget, állóképeket, hangokat, animációkat és videofilmeket továbbítani. Nyilvánvaló, hogy a nagyméretű fájlok továbbítása sokáig tart, ezért a nagyméretű fájlokat tömöríteni kell. Ismert, hogy az Interneten használt adatátviteli közegek sávszélessége alacsony. A multimédiaalkalmazások szöveget, hangokat, állóképeket, animációkat és videofilmeket tartalmazhatnak. Egy multimédiás alkalmazás legalább egy időfüggő és egy időfüggetlen médiumot tartalmaz, illetve képes ezek szinkronizált lejátszására.

A multimédiaalkalmazásokat célszerűen összeállított számítógéprendszerekben, multimédiarendszerekben lehet futtatni.

3.2 Az animáció

Az animáció különálló képekből tevődik össze. Olyan képeknek a sorozata, amelyek mozgást szimulálnak. Az animáció, olyan filmkészítési technika, amely élettelen tárgyak (többnyire bábok) vagy rajzok, ábrák stb. „kockázásával”, olyan illúziót kelt a nézőben, mintha az egymástól kismértékben eltérő képkockák sorozatából összeálló történésben a szereplők megelevenednének vagy élnének. Számítógép segítségével az animáció egyre közelebb jut a valós, életszerű lények reális mozgatásához, ugyanakkor szinte korlátlan fantáziájú háttereket lehet előállítani a virtuális ábrázoló eszközök, hang, kép, térhatás, sőt

interaktivitás segítségével. Az egymásután körülbelül 16 képkocka/másodperc vagy nagyobb sebességgel levetített animációs filmbetéteket szívesen használja a reklámszakma, az oktatás és a szórakoztatóipar. Attól függően, hogy az animáción belül milyen módszerrel készülnek a mozgások, az mindig csak rögzített képekből álló sorozat.

Az animáció megtekintésére a Test Movie parancs a legjobb. A Test Movie parancs .swf-ként menti a fájlt abba a mappába, ahol a munkafájl található. Ezután elindul a Flash Player nevű program, mely segítségével megtekinthetjük az eredményt. Az .swf fájlok a Flash-ben filmként mentett fájlok. Ez az a fájltípus, amit a weboldalakon helyezünk el. Abban különbözik a forrásfájltól, hogy nem szerkeszthető. Fontos alapelv, hogy a forrásfájlok az .fla fájlok, amelyeket meg kell őrizni. Egy .fla fájlt mindig újramenthetjük .swf-ként, de az .swf fájlból soha nem kapunk szerkeszthető .fla fájlt [3].

3.3 Alakzatok kialakítása

Kétfele módszer létezik alakzatok kialakítására: Első módszer: réteg animáció. Van egy alap képünk, ez a kiinduló helyzet, és ezt az alapképet változtatva tesszük egymás után, ezek lesznek a különböző lépések, ezekből fogjuk elkészíteni a mozgó képet. Ezeket a képeket részben átlátható fóliákra tesszük úgy, hogy az állandó részek lehetnek a fólia átlátszó részén, a változó részek pedig a fólia nem átlátszó részére kerülnek, hogy a mozgatni kívánt rész eltakarja az előző képen szereplő részt.

Második módszer: objektum animáció: Adott a teljes figura, és ezt a figurát szeretnénk mozgatni úgy, hogy részekre (pl. testrészekre), ún. objektumokra bontjuk szét. Ezen objektumok helyzetét, formáját, egyéb tulajdonságait változtatjuk úgy, hogy a mozgást érzékelhetővé tegyék. Minden mozgáshelyzetről készül egy fénykép, és ezeket egymás mögé téve, és gyorsan lejátszva látható, hogy mozog a figura.

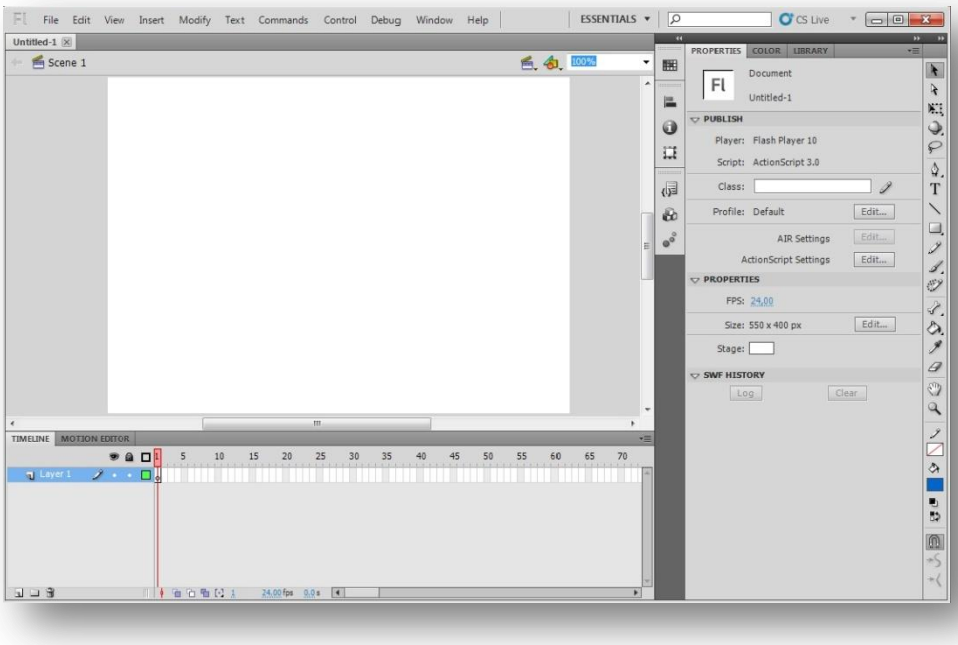
3.4 Képkockaszám

Az egyes képeket képkockáknak nevezzük. A Flash képkockái az időszalagon lévő kis téglalap alakú mezők, melyek az idővonal tetején meg vannak számozva.

A képkockaszám az a fokozat, amellyel a képkockák a felhasználó számára megjelennek, képkocka/másodpercben mérve. 24 képkocka/másodperc (frame/sec) elég a

folyamatosság látszatához (tehát 24 képet jelenítünk meg egy másodperc alatt), de néhány esetben, esetleg bizonyos filmeknél 30 kép/másodperc is előfordulhat attól függően, hogy milyen technikával dolgoznak, ill. hogy mire szeretnék azt használni. A folyamatosság látszata az előbb említett 24 kép/másodperces sebességnél már érzékelhető, hiszen az emberi szem ezt már folyamatosnak érzékeli.

Lássuk, mivel is van dolgunk igazán. Így néz ki az Adobe Flash CS5 munkaterülete (4. ábra), középen a nagy fehér négyszög a színpad, azon rajzoljuk meg a kívánt animációnkat. A színpaddal kapcsolatban nincs sok tudnivaló, egyszerűen csak egy grafikus munkafelület. Lehetőségünk van a színpadon kívüli részre is pakolni grafikákat. Ehhez nincs más teendő, csak hogy a View (Nézet) menü Pateboard (Munkaterület) előtt egy pipa legyen. Ne feledjük azonban, hogy a View menüben végzett változtatások csak azt befolyásolják, amit mi látunk, arra nincsenek hatással, ami a felhasználó számára megjelenik. Alaphelyzetben egyszerre csak egy képkocka tartalmát tekinthetjük meg. A lejátszófej csak egy képkockán állhat egyszerre, amelyet éppen nézünk. A lejátszófej kezdetben nem mozdítható, mivel csak valamilyen animáció képkockájába helyezhető, és az animációnak még csak egy kockája van.

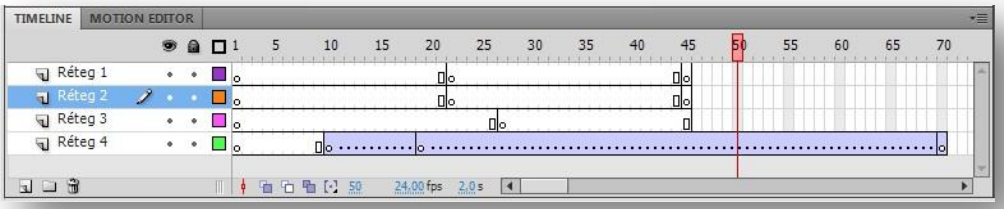


4. ábra – Adobe Flash CS5 munkafelülete
(saját képernyőmentés)

A pásztázás kifejezés sokféle animációs programban használatos – az animáció előnézetére szolgáló eljárás. Egyszerűen fogjuk meg a lejátszófejet (az időszalag kockái felett) és húzzuk előre-hátra (az animáció összes képkockáján keresztül).

Az egeret pásztázó mozdulatokkal kell használni, és innen kapta a nevét (5. ábra).

A hagyományos filmes animációkban minden kocka kulcskocka, azaz minden képkockában valami új dolog történik. Úgy adhatunk meg kulcskockákat, hogy az időszalagon (5. ábra) oda kattintunk, ahová szeretnénk elhelyezni a kulcskockát. Nem muszáj egymás után következniük. A kulcskockák között elhelyezkedő kockákban az előző kulcskocka tartalma jelenik meg, akár üres képernyő, akár kép.



5. ábra – példa az időszalag használatára

A képkockáról képkockára animációk leghasznosabb segédeszközei a hártyák. Úgy kell elképzelni, mint amikor másolópapírra rajzolunk képkockát, amelyet az előző kocka felé helyezünk. Így azon keresztül látjuk az előző képkockát, és ennek megfelelően rajzoljuk meg a következő lépést. A Flash hártyaszolgáltatása segítségével úgy szerkeszthetünk egy adott képkockát, hogy tetszőleges számú képkockát jeleníthetünk meg az aktuális képkocka előtt és után.

Valójában egy animációhoz elegendő két képkocka is. Rajzoljunk például az 1. képkockába egy egyszerű pálcikaembert, aki épp egy labdát próbál elrúgni. Majd a 10. képkockába a labdát helyezzük el a jobb oldalra, és a pálcikaember lábát nyújtsuk meg, mintha most rúgta volna el a labdát. Látható, hogy mennyire könnyen előállítottunk egy igaz egyszerű, de hatásos animációt ilyen kevés műveletből. És nem kellett mind a 10 képkockát külön megrajzolnunk. Habár egy hosszú, képkockáról képkockára animáció körülményes eljárás, látjuk majd, hogy sok csel létezik, amellyel időt spórolhatunk meg.

A kulcskockák megalkotása a mi feladatunk. Amikor úgy döntünk, hogy átmenetet készítünk, a Flash gondoskodik a kulcskockák közötti kockákról.

3.5 Átmenetkészítés

Két kulcskocka közötti rész megtartja az első kulcskocka tartalmát. Megmondhatjuk a Flash-nek, hogy a módosítást, két kulcskocka közötti változást egy átmenetkészítés nevű folyamattal egészítse ki. Az átmenetkészítés két kulcskocka kiegészítésének folyamata, amely kiegyenlíti a nagy változásokat, kis lépésekre bontva azokat. A darabos mozgás kiegyenlítődik a közbelső képkockák apró változásai által. A Flash kiszámítja ezeket az átmenetes és köztes képkockákat, tehát nem nekünk kell elvégezni a feladatot.
A mozgásátmenet szabályai:

- A kulcskockákban nem lehet több objektum.
- Az egyetlen meglévő objektumnak nem kell egyszerű alakzatnak lennie. Valójában az a legjobb, ha szimbólumpéldánnyá tesszük, mert így kisebb lesz a fájlméret.

Ha nem tartjuk be a szabályokat, a Flash figyelmeztet erre.

3.5.1 Minta követése

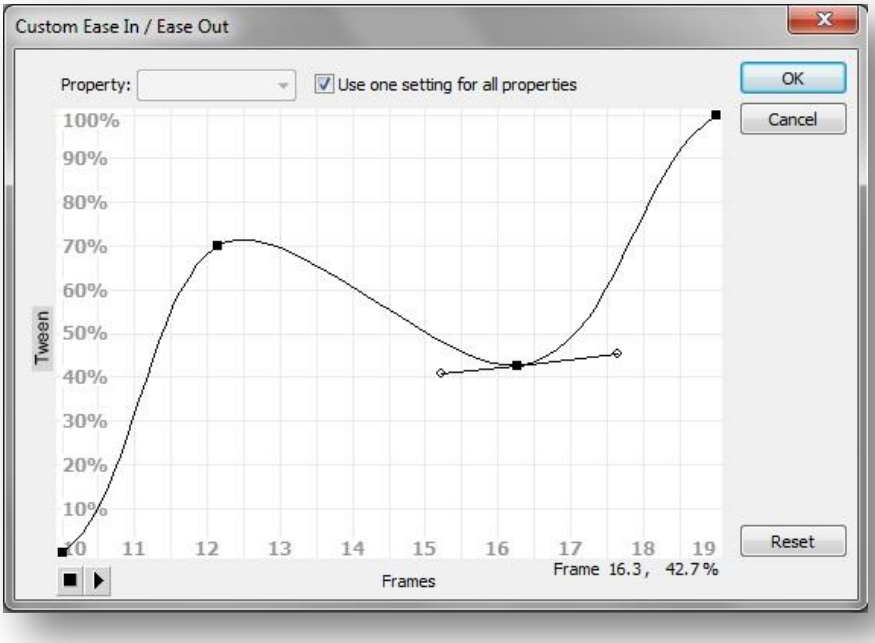
Mindig ezeket az alapvető lépéseket kell követni.

- A mozgásátmenet csak egy átmenetet használ, amelynek szimbólumnak kell lennie. Z objektumot alakítsuk szimbólummá, mielőtt elkészítjük a befejező kulcskockát, különben az új kulcskockába az első kulcskocka tartalmának másolata kerül.
- Miután létrehoztuk a kezdő és a befejező kulcskockákat, amelyek egy-egy példányt tartalmaznak, mind a kezdő, mind a befejező kulcskockán beállíthatjuk az elhelyezkedést, a méretet, az elforgatást, a ferdítést, a fényerőt, a színezetet és az áttetszőséget.
- Figyeljünk az időszalagon a lejátszófejre, hogy biztosan azon a kulcskockán álljon, amelyet szerkeszteni akarunk.

3.5.2 Mozgásátmenetek finomhangolása

Az első kulcskockában megmondjuk a Flash-nek, hogyan készítsen átmenetet a következő kulcskockához. De gyakran ugyanúgy szeretnénk befejezni a mozgásátmenetet, ahogy elkezdődött. Azt akarjuk, hogy az alakzatunk felfelé haladjon, majd vissza, lefelé, mint a jojó. Csak itt a szimbólum ugyanolyan sebességgel mozog. Ha azt szeretnénk elérni, hogy az alakzat mozgásának elején gyorsuljon, visszafelé pedig lassuljon, nem kell mást tennünk, mint az ease mezőbe írt értéket változtatni. Az erősítés, azt jelenti, hogy a mozgás lassan kezdődik, és a végére felgyorsul. A csillapítás ennek pont az ellentéte: az objektum gyorsan indul, és a mozgás végére lelassul.

De ez a beállítás még mindig csak egy átmenetre alkalmazható. A mozgásátmeneteknél (tween) az Ease In és Ease Out beállításokon túlmenően egy vektor görbén precízen beállíthatjuk a lassulás, gyorsulás mértékét. A position, rotation, scale, color és a filters külön állítható. Így olyan komplex mozgásokat készíthetünk egy Tween-el, melyeket eddig csak programozással, vagy több egymást követő Tween-el tudtunk volna megoldani. Erre jó a Custom Ease In / Ease On párbeszédablak (6. ábra).



6. ábra – mozgásátmenet erősségét finomhangolhatjuk a Custom Ease In/Ease Out párbeszédablakban

3.6 Rétegek

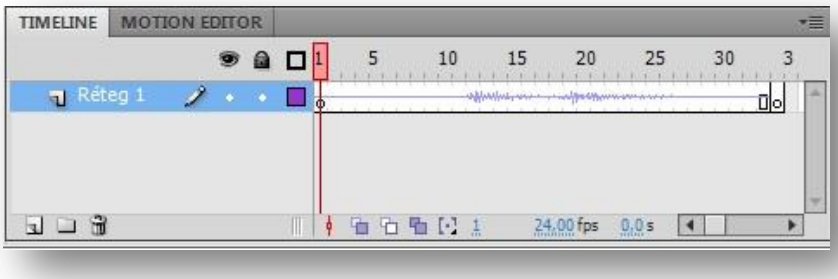
Minden egyes réteg egy-egy párhuzamos időszalag (4. ábra), ahol az animációk lejátszódhatnak. A rétegeken szereplő képek más rétegek alatt vagy fölött tárolódnak, de elsődleges céljuk, hogy különálló időszalagot biztosítsanak, amelyeken külön-külön vezérelhetjük az animációkat.

Hogyan lehet beállítani a Flash rétegeknél, hogy átlátszó legyen? Nagyon egyszerű.
A File, Publish Settings menüpontnál a HTML fület kiválasztva tudjuk a Window Mode: melletti listából kiválasztva a Transparent Windowless-t. Egy szimbólum átlátszóságát a saját properties-ében lévő "alpha"-val tudjuk állítani. (Ez a color gördülőmenüjében van).

4. Flash animáció hangja és videó fájlai

A hangok hatása gyakran tudatalatti, gyakran nem veszik észre az emberek, hogy hang fut az animáció alatt. De ha nincs hang, az rögtön feltűnik. Nagyon fontos, hogy hatékonyan használjuk, mert mindig a hangok teszik ki a közzétett filmek méretének nagy részét. Nincs lehetőség hang felvételére, csak lejátszására. Tehát a Flash-be betölthetünk hangokat, de a létrehozásukra nincs mód. Nagyon egyszerű. El kell döntenünk, melyik hangot választjuk, hol és hogyan használjuk. Támogatott hangformátumok: mp3, wav, aif, au.

Hangokat egy helyen lehet a Flash-ben alkalmazni, a kulcskockákban. Ahhoz, hogy használhassuk is, azt előbb be kell tölteni. Ezt a File, Import, Import to Library menüpontból tehetjük meg. Kiválasztjuk a nekünk megfelelő hangformátumot. De ezzel még nem történik semmi, csak be van töltve. Ezután tudjuk majd elhelyezni kulcskockához, a Properties lapon kiválasztva a megfelelő hangot (7. ábra). Vagy a színpadra húzzuk az adott hangot.



7. ábra – Hullámforma az időszalagon.

Eldönthetjük, hogy pontosan hogyan játszódjon le a hang. Minden hangpéldány számára meg kell adni a Sync beállítást. Választhatunk az Event, Start, Stop és Stream opciók közül, melyek a hang és az animáció képi elemi közötti rangsort szabályozzák. Az Event az alapértelmezett beállítás. A hangok akkor szólalnak meg, amikor elérjük az adott kulcskockát, és addig szólnak, amíg véget nem érnek. Rövid, hirtelen hangokhoz való, ez a lehetőség nyújtja a legjobb teljesítményt. A Start beállítás - ellentétben az Event beállítással - nem lehet több példánya. Elindítja a hangot, ha még nem szól. A Stop beállítást olyankor használjuk, ha azt akarjuk, hogy egy bizonyos hang elhallgasson. Úgy kell elképzelni, hogy ha az adott hang éppen szól, akkor el kell hallgatnia. A Stream beállítással a Flash kihagy néhány kockát. Folytonos hangok akkor indulnak, amikor elérjük a kulcskockát, és addig szólnak, amíg van hely az időszalagon. Összehangolja a képet a hanggal.

4.1 Hatásbeállítások

A Properties tábla számos lehetőséget nyújt a hanghatások különlegesebbé tételéhez. Az Effect (Hatás) felirat mellett lenyíló listában kiválaszthatjuk a Fade In (Fokozatos erősítés), Fade Out (Fokozatos halkítás), Fade from Left to Right (Fokozás balról jobbra) és a Fade from Right to Left (Fokozás jobbról balra) hatásokat. Hogy saját magunk határozhassuk meg a hatást a Custom (Egyedi) lehetőséget válasszuk. Az úsztatás olyan hatás, amikor úgy tűnik, mintha a hang jobbról balra vagy balról jobbra mozogna. Egyszerűen ez egy trükk. Azt csináljuk, hogy míg az egyik csatorna hangereje nő, addig a másiké csökken.

Az egyik legfontosabb dolog, amit meg kell jegyezni, hogy a kezdési- és befejezési időjelzők takarékoskodhatnak a fájlmérettel. Csak a ténylegesen felhasznált hangok és hangrészletek kerülnek bele a filmbe. A könyvtárban található használatlan hangok, valamint a hangok elejéről és végéről lecsípett darabok nem. Ha néhány másodpercet levágunk egy hang végéből, az sok másodperccel rövidebb letöltési időt is jelenthet. Továbbá a hang erejének megváltoztatása nincs hatással a fájlméretre, tehát nincs értelme a legalacsonyabb szintre állítani a szerkesztővonalakat.

A Properties táblán található egy lehetőség, amellyel meghatározhatjuk, hogy egy hang hány alkalommal ismétlődjön, vagy, hogy végtelen ciklusban játszódjon-e le. Egyes hangokat jobb eredménnyel játszhatunk le ismétlődően, mint másokat. Egy profin előkészített hang, olyan folytonosan ismétlődik, hogy amikor hallgatjuk, lehet, hogy észre sem vesszük az ismétlődést, úgy tűnik, mintha végtelen lenne.

Ha a legjobb minőségű hangot akarjuk, a fájl méret növekszik és fordítva, a kisebb fájlméret rosszabb hangminőséget jelent. Ez nagyon fontos kérdés a méret tekintetében. El kell döntenünk megéri-e a jobb minőségű hang, hogy kivárja a felhasználó a hosszú letöltési időt?

4.2 Általános közzétételi beállítások

Ahhoz, hogy egy Flash film minden alapértelmezett hangformátumát beállíthassuk, a File, Publish Settings menüpontot kell kiválasztanunk. Két különböző hangbeállítást látunk. Audio Stream (Folytonos hang) és Audio Event (Eseményhang). Az Audio Stream azokat a hangpéldányokat befolyásolja, amelyek összehangolási beállítása Stream, míg az Audio Event azokra a hangokra van hatással, amelyek vagy az Event vagy a Start összehangolási beállítást alkalmazzák. Ha Set gombra kattintunk, megjelenik az összes elérhető lehetőség. Mindkét fajta hang számára beállíthatjuk a tömörítést. Meg kell adni, hogy mekkora legyen az MP3 tömörítés, mivel minden lehetőség más-más sajátosságokkal rendelkezik, nézzük meg őket részletesen.

- Disable: Megtiltja a Flash-nek a hangok kivitelét.
- Raw: Érintetlenül hagyja a hangot, viszont meg kell adni egy mintavételezési sűrűséget (mintavételezési frekvenciát). Tesztelés közben hasznos, nem kell kivárnunk, amíg a Flash tömöríti a hangokat, akárhányszor csak a Test Movie parancsot alkalmazzuk.
- ADPCM: Majdnem ugyanaz, mint a Raw. Azzal a különbséggel, hogy más mintavételezési frekvenciát is választhatunk, mint az eredeti hangfájlé.
- MP3: Ez a legjobb minőség. Amikor közzéteszünk egy filmet, mindig válasszuk a Best Quality beállítást, mert az nem befolyásolja a fájlméretet, viszont javítja a minőséget. A Bit rate segítségével szabályozhatjuk másodpercenként mennyi adat fogadását engedélyezzük az MP3 fájlnak. Minél magasabb ez az érték, annál jobb.
- Speech Compression: Ez a beállítás az emberi hang számára optimalizált. A gyakorlatban viszont mindig össze kell hasonlítani a beszédtömörítésnek a hangminőségre és a fájlméretre gyakorolt hatását az MP3-éval, mert minden esetben más a legmegfelelőbb választás.

4.3 Videó fájlok

Volt már szó a mozgásátmenetről, alakzatokról és a hangokról. A Flash videó támogatása rendkívül kiforrottá vált. Külső videofájlokat tölthetünk be, vághatjuk és tömöríthetjük őket, és mindezt az On2 VP6 kodekjének a segítségével, ami kis fájlméret mellett is kiváló minőséget biztosít. Rengeteg módon befolyásolhatjuk a végeredményt [4].

A kodek (angolul codec) a „coder/decoder” (kódoló/dekódoló) kifejezésre utaló, dupla értelemmel bíró szó, amely egy adat- vagy jelfolyam átalakítására szolgáló eszközt vagy programot fed le. Egy kodek átalakíthat egy adat- vagy jelfolyamot egy kódolt formátummá (gyakran átviteli, tárolási, vagy rejtjelezési célból), ugyanakkor képes dekódolni is azt a formátumot, egy akár megtekintésre, akár kezelésre jobban alkalmas formátummá.

Példának okáért számos multimédiás adatfolyam kell, hogy tartalmazzon együtt hangot és képet, de gyakran még a kettő szinkronizálására szolgáló „metaadat”-ot is. E három adathalmaz mindegyikét kezelhetik egymástól független programok, eljárások, vagy hardware eszközök; de a multimédiás folyamatos adatátvitel szemszögéből előnyösebb, ha ezek egyetlen egységbe vannak zárva. A hang vagy videó adatok nyers kódolt formáját nevezik esszenciának ("essence"). Megkülönböztetésként a metadata információs tartalomtól, amellyel együttesen alkotják magának a "stream"-nek az információs tartalmát. Ehhez a csoportosításhoz még egy ún. "wrapper" (csomagoló) is járulhat, ami elősegíti a stream-hez való hozzáférést és annak robusztusságát. A kodek nem keverendő össze a videófájl-formátummal, ami a kodek által kódolt hang/kép információ tárolására szolgál. A leggyakoribb hang/kép fájlformátumok, mint például ".ogg", ".mpg", ".avi", ".mov" stb. egy, de akár több különböző kodekkel kódolt információt is tárolhat. (Például ".avi" lehet DivX, de akár XviD is.) Ha videó közvetítést akarunk készíteni, két fontos dolgot kell eldöntenünk:

- melyik videó fájl formátumot fogjuk használni?
- és milyen módszerrel fogjuk a videót közvetíteni?

4.4 Videók beágyazása és külső videók lejátszása

Kétféleképpen játszhatunk le videókat a Flash-ben. Vagy beágyazzuk azokat, ilyenkor be kell töltenünk, tömörítenünk kell, majd el kell helyeznünk a videót a fő fájlban, vagy .flv fájlokba tömörítjük őket és külső fájlként játsszuk le őket. A videók beágyazása nem csak megnöveli a fájlméretet, hanem a minőség sem lesz olyan jó, mintha külső .flv fájlt játszanánk le. Ennek két oka van: nem használhatjuk a VP6 kodeket, illetve a hosszabb videókban a hang sem marad összhangban a képpel. Általában az a legjobb megoldás, ha egy .flv fájlt hozunk létre, amelyet külső fájlként játszunk le. Viszont vannak olyan esetek, amikor a beágyazás mégis a jobb választás: amikor a videó nagyon rövid, és nem szeretnénk szétosztani több fájlba a tartalmat, és ha képkockaszintű vezérlésre van szükségünk. A külső .flv fájlokat csak kulcskockákon állíthatjuk meg. Ha azt szeretnénk, hogy a felhasználók kikockázhassanak egy videóklippet, a videó beágyazása mellett célszerű döntenünk. Függetlenül attól, hogy a végső videót beágyazzuk, vagy külső fájlként kívánjuk lejátszani, a videót mindig tömörítenünk kell.

4.4.1 Beágyazott videó lejátszása

Egy kulcskockához rendelt, Event (Esemény) beállítású hang lejátszása akkor indul el, ha elérjük a képkockát. És addig tart, amíg véget nem ér, még akkor is, ha az időszalag lejátszása lelassul, hogy megjelenhessen az összes képkocka. A hangfolyamok ezzel szemben rögzítve vannak az időszalagon. Tehát gondoskodnunk kell róla, hogy elegendő idő álljon rendelkezésre a hang lejátszásához. Egy kulcskockába helyezett beágyazott videónál is elegendő helyre van szükség az időszalagon, hogy a hangsávot végig lejátszhassuk. Ha például van egy 10 másodperc hosszú videónk, aminek a képkockaszáma 12 képkocka másodpercenként, akkor 120 képkockára van szükségünk az időszalagon. A Flash pontosan megmondja, hogy hány képkockára van szükség. A videók a szimbólumtípusok közül a grafikaszimbólumokra hasonlítanak leginkább. Például előzetes képet kaphatunk a videóról, ha végighúzzuk a lejátszófejet az időszalagon, amit meg is kell hosszabbítanunk, hogy ráférjen a videó összes képkockájára [5].

4.4.2 Külső .flv videó fájlok lejátszása

A Flash képes külső .flv fájlok lejátszására, de ezeket előbb létre kell hozni. Alapvetően négy módszer létezik:

1. Betöltünk egy videót a Flash-be, és az első három közzétételi beállítás közül választunk. Az .flv fájl létrehozása mellett a Flash egy összetevőt is beállít.
2. Külső alkalmazás segítségével készítjük el az .flv fájlt, például az Adobe Flash Video Encoder On2 Flix nevű programja, vagy a Sorenson Media Squeeze for Flash segítségével.
3. Közvetlenül egy támogatott videó szerkesztőben hozzuk létre a filmet egy olyan számítógépen, amelyen telepítve rendelkezésre áll az Adobe Flash Video Encoder.
4. Beágyazzuk a videót, majd a Könyvtárban lévő videó elemből .flv fájlt készítünk. Bár ez eredményezi a leggyengébb minőséget.

Mindegy hogyan készítjük el a fájlt, mindenképpen a külső .flv fájlok lejátszása adja a legjobb eredményt, a kép és a hang összhangban marad. A külső fájlok lejátszása azonban egy kicsit több munkával jár. Először is fel kell töltenünk mind a .swf, mind a .flv állományt. Ha beágyazzuk a videót, csak a videó képkockáiról kapunk előképet. Ha animációt szeretnénk a videóra rajzolni, beágyazott videót kell használnunk.

4.4.3 Egyszerű .flv videó készítése Flash-ben

Nyissunk egy új dokumentumot Flash-ben. Körülbelül 400x400 pixel nagyságúra állítsuk a színpadot, ebből 400x300 lesz a video lejátszó kijelző része, a maradék területre pedig play, stop, és egy visszalépés gombot teszünk majd. Fontos megemlíteni, hogy a NetStream osztály AS2 specifikus!

A rétegek: AS, keret, video, play és stop gomb, s még egy gomb, amivel visszaléphetünk majd a film elejére. A keret rétegre egy 400x300-as tetszőleges színű körvonal kerül, csak hogy tudjuk hol fog megjelenni a videónk. A video rétegen egy embed videót helyezünk el, szintén 400x300-as méretben. A library (CTRL+L) jobb felső sarkában lévő ikonra kattintva a legördülő listából válasszuk ki a "New Video" részt. Ekkor a library-ban létrejön egy beágyazott üres video, ezt húzzuk ki a színpadra, úgy hogy a kerethez

passzoljon, a properties panelen pedig lássuk el a tv_mc instance névvel. A gomb rétegekre csak egy-egy gomb kerül, amivel majd a mozinkat minimális mértékben irányíthatjuk. A gombokat ugyanúgy, mint a videónál a legördülő menüből a ”New Button” részt választjuk, és a felhasznált instance nevek a következők: play_stop_btn, elejere_btn.

Ezek után lépjünk az AS layer-re és gépeljük a következő pár sort, elég minimális tudással rendelkező player-t készítettünk, szerintem különösebb magyarázatra nem szorul ez a pár sor:

```
var nc:NetConnection = new NetConnection();
nc.connect(null);
var ns:NetStream = new NetStream(nc);
kijelzo_mc.attachVideo(ns);
ns.play("video.flv");
play_stop_btn.onPress = function()
{
        ns.pause();
        };
elejere_btn.onPress = function()
{
        ns.seek(0);
        };
```

Tehát röviden: Új kapcsolat létrehozása, a videó csatolása a kijelzőnkhöz, majd elindítjuk a videót, egy gombbal szabályozzuk a lejátszást, egy másik gombbal pedig visszalépünk a film elejére.

5. ActionScript

Az ActionScript egy ECMAScripten alapuló programozási nyelv (Az ECMAScript programozási nyelv, melyet főként webes alkalmazásokra fejlesztettek ki. A nyelv leírása ECMA szabvány), mely nagyban hasonlít a széles körben elterjedt JavaScriptre. Az ActionScript elsősorban az Adobe Flash objektumok programozásához készültek. Jelenleg több verziója is létezik, ezek közül a legfrissebb a 3.0.

A Flash CS5 scriptszerkesztőjét funkcionalitásában továbbfejlesztették, vizuális felhasználói felülettel látták el, amelynek köszönhetően jelentősen könnyebbé vált a scriptek írása. A beépített Script Assist segít boldogulni az ActionScript nyelv lehetőségeivel. Az idősávon készített (nem védett) animációkat átkonvertálhatjuk ActionScript kóddá. Ennek köszönhetően a kód könnyedén újrafelhasználhatóvá és szerkeszthetővé válik. Az újra felhasználható modulokkal jelentős idő takarítható meg a fejlesztések során.

Az ActionScript első parancssorai még meglehetősen primitívek és egyszerűek voltak, hiszen elsősorban médialejátszásra akarták használni. Így az első parancsok, a play(); a stop(); a goto(); és hasonlók voltak. Azóta azonban túlnőtte az eredeti célkitűzést, és ma már egész weboldalakat készítenek a Flash és az ActionScript segítségével. Az ActionScript alkalmas az oldalon a látványos dolgok produkálására is, de ezen felül manapság képes együttműködni különböző adatbázisokkal (például MySQL) és különböző programnyelvekkel is (például PHP vagy JavaScript).

A Flash folyamatosan bővül a mai napig is, de még sosem ment át akkora fejlődésen, mint az új ActionScript esetén. Teljesen újraírták, és a futtatást egy másik virtuális gép végzi a Flash player-en belül. ActionScript nélkül a Flash film minden egyes alkalommal ugyanúgy játszódik le. Segítségével viszont megtehetjük, hogy a filmet megállítsuk vagy elindítsuk. Ma már igazán hatékony nyelvvé nőtte ki magát. Ez azt is jelenti, hogy a nyelv sokkal bonyolultabb. Parancsfájlainkat vagy pontosan megírjuk vagy nem fognak rendesen működni. Nézzük meg röviden a korábbi verziókat és jellemzőit:

ActionScript 1.0 - Minden egyes nyelvi elem objektum volt és a származtatást prototípus objektumok segítségével valósították meg. A prototípusok mindig a származtatott objektumra mutattak, és ha egy metódust nem talált a player végigjárta a prototípusokat addig, amíg meg nem találta, így valósult meg a származtatás. Másik fontos dolog hogy nem támogatta az osztály definíciókat nem volt lehetőség a class kulcsszó használatával osztályok

létrehozására. Erre a célra függvényeket használtak, amik template-ként szolgáltak egy objektum elkészítéséhez.

ActionScript 2.0 - Tulajdonképpen csak egy feltunningolt AS1 mivel a nyelv alap működésében nem hozott semmi újat csak a fejlesztők dolgát könnyítette meg olyan újdonságokkal, mint a fordításidejű típus ellenőrzés vagy a class kulcsszó bevezetése, amivel lehetőség nyílt más nyelvekhez hasonló osztályok létrehozására. Megjelentek az interface-ek is, amik szintén a fejlesztők dolgát voltak hivatottak megkönnyíteni. A belső maradt ugyanaz, az osztályok példányosításuk után objektumokká váltak és elvesztették a láthatósági információkat, amit a private vagy public kulcsszavak használatával lehetett megadni. Hiába lehetett típusokat megadni, ez csak a fordításkor tudta kiszűrni a hibákat, mivel nem volt futás idejű típus ellenőrzés.

ActionScript 3.0 – Külön virtuális gépet kapott, amit AVM2-nek (Actionscript Virtual Maschine 2) hívnak. Ennek segítségével, sokkal gyorsabban és megbízhatóbban futtatható Flash alkalmazások készíthetőek. Megnőtt kódok futtatásának sebessége is, helyenként 10x gyorsabban fut ugyanaz a kód AVM2 alatt. ActionScript 2.0-ban csak 1 fajta szám létezett, mégpedig a *Number*. Ebben a típusba ugyanúgy bele kell férnie a lebegőpontos és negatív számoknak egyaránt. Ezért került bevezetésre két új típus az *int* (integer) és az *uint* (unsigned integer). Az *int* egész számok tárolására alkalmas, pozitív és negatív irányban is. Tehát csak egész számokkal végzendő számítások során vagy ciklusok írása során célszerű használni. Az *uint* pozitív egész számok tárolására alkalmas, negatív számok tárolására nem használható [6].

AS3-ban a típus információk futás időben is megmaradnak így a player értékadásnál ellenőrizni, tudja a típusok kompatibilitását, és eltérés esetén megpróbálhatja átkonvertálni vagy hibát dobhat, ami nagy segítséget nyújt a fejlesztők számára. A debug flash player minden hibáról részletes értesítést küld, amiben látható a hiba leírása és a sorszám, ahol a hiba keletkezett. Így el se lehetne képzelni kivételkezelés nélkül.

A namespace-ek deklarálása ismerős lehet azoknak, akik már dolgoztak más komolyabb programozási nyelvekkel. A namespace azt határozza meg, hogy adott változót vagy metódust honnan lehet elérni.(Ilyenek például: private, public, protected). Lehetőségünk van saját láthatóságot definiálni, ami megaakadályozza a névütközést egy osztályon belüli azonos nevű metódusok deklarálása esetén, vagy akár xml kezelésnél is hasznosnak bizonyulhatnak

Most már minden változó deklarálásakor kötelezően ki kell tenni a *var* kulcsszót, ha nem így tennénk fordítási hiba. Típust is adhatunk a változónak:

var myVar:String = ”Diplomamunka”;

Lehetőségünk van konstansok deklarációjára is. A konstansok olyan változók, amiknek csak egyszer lehet értéket adni majd nincs lehetőség a megváltoztatásukra.

A *const* kulcsszó változó név elé írásával hozhatunk létre konstansokat: *const CONSTANT:String = "konstans";*

A többi nyelvhez hasonlóan csupa nagybetűvel szokás írni a konstansokat, hogy a többi változótól meg lehessen különböztetni őket [7].

Ha deklaráláskor nem adunk értéket a változónak, akkor az AS3-ban minden típusos változó kap egy undefinied-tól különböző alap értéket. Változótól függően más és más érték lehet.

Ha egy változónak nem adunk típust, akkor a fordító típustalannak fogja értelmezni, az alap érték undefined lesz és ezután akármilyen értéket rakhatunk bele, ami lehet szám vagy szöveg is. Ezeket az alábbi táblázat foglalja össze:

Típus	Alap Érték
Boolean	false
int	0
Number	NaN
Object	null
String	null
uint	0
Típustalan változók	undefined
Más osztályok	null

Forrás: http://www.devnet.hu/flash/tutorials/actionscript/as-as3intro/

AS3 a primitív típusokat automatikusan konvertálja, ha nincs típusegyezés. Ez azt jelenti, hogyha mi például egy *int*-et szeretnék elhelyezni egy *boolean* típusban, nem kapunk hibát legfeljebb egy figyelmeztetést CS5-ban, ami közli velünk, hogy konverzió fog történni.

A függvényeket a *function* kulcsszó segítségével deklarálhatunk. A függvény neve bármi lehet, viszont nem kezdődhet számmal és nem lehetnek benne szóközök. De a lényegi rész a kapcsos zárójelek között van. Ide bármilyen egy vagy többsoros kódot írhatunk.

function myFunction(myVar:Object):void{

}

A függvények is objektumok, ami azt jelenti, hogy nem maga a függvény másolódik, hanem a referencia adódik át. A korább verziókhoz képest, ami változott, hogy most már a függvénynek is lehet *length* tulajdonsága, ami a benne szereplő paraméterek számát adja meg. Ha megadjuk a várt paraméter típusát, akkor fordítási időben ellenőrzi, hogy megfelelő típussal hívtuk-e meg, és természetesen megadható a visszatérési típus is. Primitív típusok érték szerint adódnak át, míg referencia típusok referenciaként. A függvényeket használhatjuk változóként is, így egy változó lehet *function* típusú, ami a beépített típusok közé tartozik. Így egy változóhoz értékadással hozzá lehet rendelni egy már definiált függvényt.

Lehetőségünk van más módon bejárni az objektum elemeit. Ez az újfajta szintaxis megkönnyíti a dolginkat, ha csak az objektum által tárolt értékekre vagyunk kíváncsiak, mivel nem a *String* kulcsot, hanem rögtön az objektumban tárolt értékeket adja vissza.

```
for each(var val:* in myObject){
    trace(val)
}
```

A ”val” változóba kerülnek az értékek, így a megadott típusnak egyeznie kell az objektumban tárolt adatok típusával. Egy változót szándékosan típustalannak deklarálhatunk a * segítségével.

Essen néhány szó a változásokról:

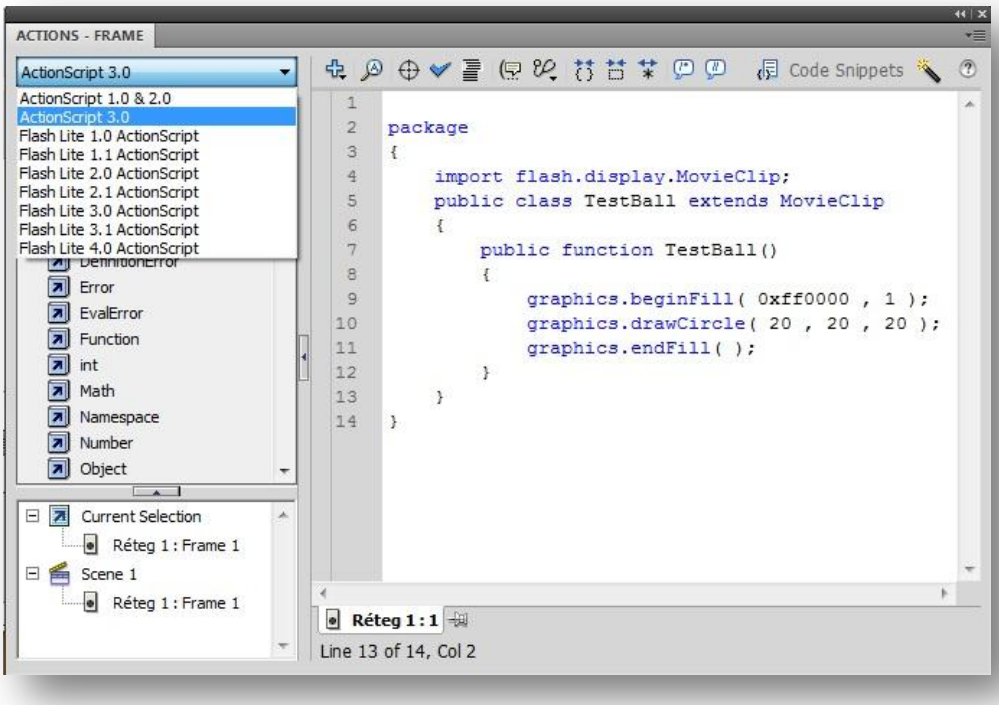
AS2	AS3
_x	x
_y	y
_width	width
_height	height
_xscale	scaleX (0-1)
_yscale	scaleY (0-1)
_xmouse	mouseX
_ymouse	mouseY
_alpha	alpha (0-1)
_visible	visible

Forrás: http://www.devnet.hu/flash/tutorials/actionscript/as-as3intro/

Ha egy objektumot ki akarunk törölni, le kell nulláznunk az arra mutató összes referenciát, és akkor a Garbage Collector fogja felszabadítani az objektumhoz rendelt tárterületet a memóriából [8].

Az ActionScriptbe számtalan eseménykezelőt építettek. Tulajdonképpen az eseménykezelés alatt arról van szó, hogy hogyan írhatunk olyan kódot, ami különféle eseményekre reagál. Ilyen esemény, például amikor egy felhasználó az objektum felett mozgatja az egeret, egy gombra rákattint, az egérgomb lenyomása és felengedése, különféle objektumok húzása és dobása. ActionScriptben gondolkodni nem könnyű, ezért először mindig azt kell megfogalmaznunk, mit szeretnénk elérni. Intelligenciával ruházzuk fel az objektumokat, hogy a felhasználó beavatkozásaira reagálni tudjanak. Az ActionScriptet hozzárendelhetjük képkockákhoz, moziklipekhez és gombokhoz.

Az ActionScript kódot mindig az Actions táblába gépeljük be, melyet az F9 billentyűvel érhetünk el leghamarabb. (8. ábra)



8. ábra – példa Actions tábla használatára

Mit is látunk a példán? Létrehoztunk egy csomagot (package) ami tartalmaz egy publikus (kívülről elérhető) osztályt. Egy csomag több osztályt is tartalmazhat, de csak egy publikust, a többit a package-et lezáró kapcsos zárójel után írhatjuk be, és ezek az osztályok csak a lokális csomagban elérhetőek. Aztán látunk olyat is, hogy public class TestBall extends MovieClip. Ez azt jelenti, hogy a TestBall osztályunk a MovieClip osztályból lett származtatva, hiszen látni is szeretnénk valamit futtatás után. Ami nagy újítása az AS3-nak, hogy már nem csak a MovieClip osztály az egyetlen, ami a megjelenítéshez kapcsolódik, ott van még a Shape és a Sprite is, nekik nincs Timeline-uk, így jobban lehet optimalizálni velük.

6. Flash Builder 4

Az Adobe Flash Builder 4 (korábban Flex Builder néven volt ismert) használatával a szoftverfejlesztők hatékonyan fejleszthetnek platform gazdag internetes alkalmazásokat (RIA) és tartalmakat a nyílt forráskódú Flex-keretrendszerben. A szoftver a fejlesztés felgyorsítására kínál szolgáltatásokat. A fejlesztést olyan eszközök könnyítik meg, mint a hibakereső és tesztelő funkció, az intelligens kódolás, illetve a felhasználói felület vizuális grafikai tervezése.

- Termelékeny, Eclipse™-alapú IDE segítségével fejleszthetünk, amelyben szerkeszthető MXML, ActionScript® és CSS, továbbá tartalmaz kódszínező, utasításkiegészítő, kódösszecsukó szolgáltatást, valamint lehetővé teszi az interaktív, lépésenkénti hibakeresést és általános kód automatizált készítését, emellett importálhatók grafikai elemek az Adobe Creative Suite® csomag elemeiből is: például az Adobe Flash Professional, Illustrator®, Photoshop® és Fireworks® alkalmazásokból is.
- A felhasznált memóriaméret és processzorteljesítmény megfigyelését és elemzését elvégző profilelemző eszközök használatával az alkalmazás működése felgyorsítható.
- Automatizálható a fejlesztési folyamat az új parancssorépítő képességnek köszönhetően.
- Megjegyzéseket jeleníthetünk meg az MXML- és ActionScript-szerkesztőkben az új ASDoc segítségével.
- Az alkalmazás megjelenítése testreszabható a CSS és a grafikai tulajdonságok módosításával. Grafikai nézetben gyorsan beállíthatók a leggyakoribb tulajdonságok, és előnézhető az eredmény.A Flex-témák könnyen tallózhatók és importálhatók, majd alkalmazhatók a projektben.
- A Flash Builder 4 alkalmas Adobe AIR® alkalmazások készítésére, hiszen tartalmaz minden eszközt, amely az AIR alkalmazások fejlesztéséhez, hibakereséséhez, csomagolásához és szignálásához szükséges. Gazdag internetes alkalmazások (RIA) gyorsan fejleszthetők az Adobe AIR segítségével, és futtathatók asztali alkalmazásként is azonos képességekkel és kódrendszerrel, mint az internetböngészőben.

7. Photoshop CS5

Úgy gondolom, nem mehetek el az Adobe rengeteg újdonsággal rendelkező tagja, a Photoshop mellett. Nem járok messze az igazságtól, ha azt állítom, hogy a Photoshop neve mára teljesen összefonódott a számítógépes képszerkesztéssel.

A Photoshop története valamikor 1988 nyarán kezdődött. Sokáig csak Apple Macintoshon futtatható programként volt jelen. Míg meg nem jelent a 2.5-ös kiadás, ami már Windows gépeken is működött. Ezt követő időszakokban szinte kétévente jelentek meg verziófrissítések. 2003-ban a PhotoShop 7.0 megjelenésével az Adobe úgy döntött, hogy átkereszteli az alkalmazást. Innentől kezdve már Photoshop CS, vagyis PhotoShop Creative Suite néven futott. Ma már ez a rövidítés inkább egy designer csomagot jelent. Kiemelkedett a Photoshop mögül és továbbment, kiegészítve további hasznos szoftverekkel. A jelenlegi verzió a CS 5, rengeteg újítást pakoltak bele, és én ezek közül csak néhányat említek. A felület az elődjéhez képest nem sokat változott. A fejlesztők maradtak a jól megszokott kinézeténél.

- Gyorsabb lett a munka a 64 bit-es támogatásnak és az új Mercury Playback motornak köszönhetően. A kiterjesztett 64 bites támogatás támogatja az Adobe After Effects, Adobe Premier Pro, és Photoshop programok jobb rendszermemória kihasználását, tehát 10-szer gyorsabb munkát tesz lehetővé magas felbontás használata mellett.
- Megváltozott, okosabb lett a kijelölő eszköz is. A legnehezebb dolgunk akkor adódik, ha mondjuk egy szőrös kutyát, vagy egy gazdag hajkoronával megáldott fejet szeretnénk kijelölni. Ilyen esetekben sokat segít az intelligens Refine Edge funkció. Ha nagyjából kijelöltük az arra érdemes területet és rányomunk a Refine Edge-re, megjelenik a vadonatúj Smart Radius opció, amely minden korábbinál pontosabb kijelölést tesz lehetővé.
- Ugyancsak újdonság a Content-Aware Fill (tartalomfüggő kitöltés). Ha van egy olyan képünk, melyről a lehető legegyszerűbben szeretnénk leretusálni egy kézben tartott tárgyat, vagy mondjuk egy kőfal előtt álló személyt, akkor elég csupán nagyjából kijelölnünk az adott objektumot, a Content-Aware Fill pedig gyakorlatilag teljesen magától elvégzi a retusálást. Az igazsághoz tartozik, hogy a legjobb végeredmény elérése

érdekében adott esetben szükség lehet némi kézi kiigazításra is, de ettől függetlenül ez az új funkció nagyon is jól használható.

- Az egyik leglátványosabb új funkció az úgynevezett Puppet Warp. Betöltünk egy virágot ábrázoló fényképet, a kis növényen definiálunk néhány fix pontot, majd pedig gyakorlatilag úgy deformálhatjuk, hajlítgathatjuk a virágot, ahogy csak szeretnénk. Az animációs programok csontvázaihoz hasonlóan kulcspontokat jelölhetünk ki egy képen, majd ezek mentén kényünk-kedvünk szerint alakíthatjuk át a képrészletet.
- Az Adobe Photoshop CS5 Extended mindazt tartalmazza, amit a Photoshop CS5, emellett 3D és mozgást megjelenítő tartalom szerkesztésére alkalmas eszközökkel is rendelkezik. Az új Adobe Repoussé technológiával könnyen készíthetünk 3D emblémákat és rajzelemeket bármilyen szövegrétegből, alakzatból, vagy maszkból, emellett csavarhatjuk, forgathatjuk és domboríthatjuk is a terveket, a különféle vizuális hatások elérése érdekében. A 3D objektumok valóságos megjelenésének ábrázolására az új anyagtár a lehetőségek gazdag kínálatát nyújtja, amelyek között króm, üveg és parafa is található. Az új árnyékfogóval gyorsan és egyszerűen tudunk árnyékokat létrehozni, ami által tovább tökéletesíthető a 3D objektumok megjelenése. Mozgó tartalom esetén az olyan eszközök, mint a festés, a szöveg vagy a klónozás könnyen, egyetlen gyorsbillentyűvel közvetlenül használhatók egy videó több képkockáján.

Mindent egybevetve az új Photoshop nagyon jól sikerült. Ha elég memóriánk van (több mint 4 GB), szembetűnő gyorsulást érhetünk el, habár én kevesebb memóriával és magas felbontással is gyorsabb működést tudtam elérni, mint a korábbi verzióknál tapasztaltam.

Összefoglalás

A diplomamunka megírásakor arra törekedtem, hogy olyan témaköröket fejtsek ki jobban és világosítsak meg, amelyeket mind az egyetemen, mind könyvekben csak részben vagy egyáltalán nem említettek eddig.

Az első két fejezetben a Flash kialakulásáról, és magáról a technológiáról írtam. Ennek ismertetése elengedhetetlen ahhoz, hogy továbblépjünk a következő fejezetekre, ahol már a Flash konkrét eszközeit tárom fel a kedves olvasó előtt. Különös tekintettel részletesen kitértem a Flash multimédiás eszközeire, a hang és videó beállításokra.

Az ötödik fejezetben részletesen foglalkoztam a Flash legfrissebb programozási nyelvével. Ismertetettem az ActionScript 3.0 scriptnyelv újításait, amivel még könnyebben lehet hatásos tartalmakat létrehozni. Ennek segítségével érhetjük el a kitűzött célunkat, és könnyebben készíthetünk hatásos és figyelemfelkeltő animációkat.

A diplomamunkámban feltételeztem az olvasó alapszintű Flash ismereteit, ennek ellenére dolgozatom elején felvázoltam az alapokat. Bővebb ismereteket az irodalomjegyzékben felsorolt könyvekben és linkeken lehet találni.

Összegezve, a Flash technológia nagyon gyorsan fejlődik. Úgy gondolom, nagy jövő áll előtte. A CS5 csomagban található eszközökkel együtt kiváló fejlesztő eszköznek bizonyul. Nem kell programozónak lenni ahhoz, hogy kis Flash videókat gyártsunk. Persze komolyabb alkotásokhoz elengedhetetlen legalább egy közepes ActionScript programozási ismeret. Úgy gondolom, sikerült bemutatnom a technológia multimédiás lehetőségeit a lehető legpontosabban. Sokrétűsége miatt csak ajánlani tudom mindenkinek.

Irodalomjegyzék

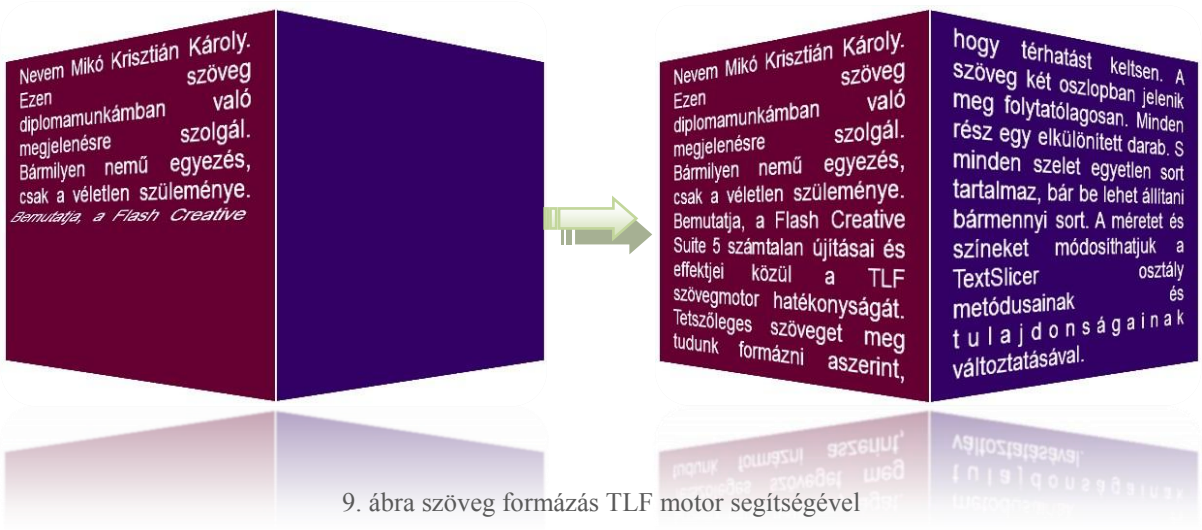
1.  Phillip Kerman - Tanuljuk meg az Adobe Flash CS3 Professional használatát 24 óra alatt, Kiskapu Kiadó, 2007. [162-173, 201, 386-388]
2.  Adobe Flash Professional CS5 - http://cs5.hu/info/adobe-flash-pro/
3.  Adobe Creative Team - Adobe Flash CS3 Professional. Tanfolyam a könyvben, Perfact Kiadó, 2008.
4.  Videó készítés - http://www.mediacollege.com/video/streaming/overview.html
5.  Dr. Dobb's Journal 2002, april #335 San Francisco, CMP Media CCC. Globalized Web Application & ASP.NET[67.-72.] and The MPEG-4 JAVA API & MPEGlets[40.-45.]
6.  Colin Moock - ActionScript 3.0 a gyakorlatban. Az ActionScript-programozás alapjai, Kiskapu Kiadó, 2008.
7.  ActionScript - http://www.devnet.hu/flash/tutorials/actionscript/as-as3intro/
8.  Macromedia Flash MX gyorstalpaló - http://edutech.elte.hu/jegyzet/flash/flash-000.html
9.  Robert Reinhardt, Jon Warren Lentz - Flash 5 Biblia I-II kötet. Kiskapu Kiadó, 2001.
10.  Hivatalos Adobe Flash honlap - http://www.adobe.com/products/flash/

Függelék

Dupla hasábos szöveg térhatású megjelenésének elérése az új TLF szövegmotor segítségével az Adobe Flash Creative Suite 5 programban

Egy izgalmas példát mutatok be, amely első ránézésre viszonylag bonyolultnak és nehezen emészthetőnek tűnik, de minden lépését megmagyarázom, hogy könnyen érthető legyen.

A szöveg futásidőben töltődik be dinamikusan. Köszönhetően az új szövegmotor (Text Layout Framework) használatának, a szöveget 2 részre osztja a program. Úgy látszik, mintha térben elhelyezkedő kocka két oldalán helyeztük volna el a kért szöveget. Minden sor 3 dimenzióban jelenik meg egy úgynevezett “lenyíló” effekt kíséretében. Ezért nagyon jó megjelenést kölcsönöz irományunknak (9. ábra).



9. ábra szöveg formázás TLF motor segítségével

A TextSlicer ActionScript3 osztály:

A TextSclicer osztály felelős a megjelenítésért. Tehát ezen osztály húzódik meg a térhatás mögött. Az osztály darabolja szeletekre a szöveget és helyez minden szeletet egy különálló Sprite tárolóba. Minden ilyen tároló egy ContainerController példánya, ugyanúgy a TextFlow osztály példánya. Ennélfogva, a szöveg fel van osztva, de terjedelme korlátozott. Ahhoz, hogy a TLF szövegünket létrehozzuk és manipuláljuk, szükséges betöltenünk az AS3 osztályokat a flashx.textLayout csomagból.

A konstruktor beállítja a szöveg szélességét, más szóval minden szelet alapértelmezett szélességét (default: 240):

new TextSlicer(sw:Number=240)

Tulajdonságok: Minden változó neve magában hordozza tulajdonságát.

instance.fontColor

Szám. A szín hexadecimális száma. Az alapértelmezett a fekete.

instance.fontSize

Szám. Az alapértelmezett értéke a 14px.

instance.vertPadding

Szám. Beállítja a sortávolságot. Alapértelmezett a 2px.

instance.hasBack

Logikai. Alapértelmezett értéke a hamis. Meghatározza a szöveg hátterét.

instance.backColor

Szám. Szín hexadecimális értéke. Alapértelmezett a fehér.

instance.backAlpha

Szám. Alapértelmezett az 1.

instance.textAlignment

Szöveg. Alapértelmezett a balra. Lehetséges értékek: balra, jobbra, középre, sorkizárt.

instance.numLinesPerSlice

Egész. Egy szelet hány sort tartalmazzon. Alapértelmezett az 1 sor.

instance.sliceHeight

Csak olvasható. A szeletek magassága.

instance.sliceWidth

Csak olvasható. A szeletek szélessége.

instance.importText(t:String,m:String)

Az első paramétere t, a szöveg, amit meg kell formázni. A második paraméter m, a szöveg típusa. Amely lehet html, textlayout vagy plaintext. Ebben a példában a textlayout típust használom, ami nem más, mint a TLF.

```
import flashx.textLayout.container.ContainerController;
import flashx.textLayout.elements.TextFlow;
import flashx.textLayout.conversion.TextConverter;
```

Létrehoztam egy URLLoader-t hogy betöltse a text fájlt amit könnyedén lehet szerkeszteni bármikor. Nevezzük most ezt szoveg.txt-nek.

```
var ldr:URLLoader = new URLLoader();
ldr.addEventListener(Event.COMPLETE, loadComplete);
ldr.addEventListener(IOErrorEvent.IO  ERROR. loadError);
```

Majd definiáltam egy String típusú változót a betöltött text fájl adatainak tárolására.

```
var fileContent:String;
```

Példányosítottam a TextFlow osztályt (“flow”). A TextFlow objektum fog olvasni információkat a felületről és fogja megformázni a szöveget helyesen megjelenítve azt.

```
var flow:TextFlow = new TextFlow();
```

Betöltöm a text fájlt. Ha a fájl ugyanabban a könyvtárban van, mint a html és swf fájlok, és persze a megfelelő fla fájl, akkor a text fájl neve lesz a cím.

```
ldr.load(new URLRequest("szoveg.txt"));
txtOutput.text="Loading an external file...";
function loadError(e:IOErrorEvent):void {

        txtOutput.text = "Error loading an external file. The server may be busy. Try refreshing
        the page.";
}
```

Mikor a betöltés kész, meghívom az initText függvényt.

```
function loadComplete(e:Event):void {

fileContent = ldr.data;
txtOutput.text="An external text file, szoveg.txt, was loaded as the movie opened. The
text and the information about layout and formatting are contained in the text file.";
ldr.removeEventListener(Event.COMPLETE, loadComplete);
ldr.removeEventListener(IOErrorEvent.IO_ERROR, loadError);
initText();
}
```

Az initText függvény a TextConverter osztály importToFlow statikus metódusát használja. Importálja az adatokat a szoveg.txt fájlból a „flow”-ba. Ezt követően, a flowComposer függvényt használom a táblánk létrehozására, amelyben megjelenítem a szöveget. Beállítom a méretét is (570x370). Az updateAllControllers függvény teszi lehetővé, hogy mindent frissíteni tudjunk.

```
function initText():void {
    flow = TextConverter.importToFlow(fileContent,TextConverter.TEXT_LAYOUT_FORMAT);
    flow.flowComposer.addController(new ContainerController(container, 570, 370));
    flow.flowComposer.updateAllControllers();
}
```

Ha egy szöveges fájl helyett XML fájlt szeretnénk betölteni, azt kevés változtatással is meg lehet valósítani:

```
    var fileContent XML;
```

és az initText függvényben:

```
    flow=TextConverter.importToFlow(fileContent,TextConverter.TEXT_LAYOUT_FORMAT);
```

Mivel az importToFlow metódus első argumentuma objektum, így String vagy XML objektum is lehet. A TextConverter osztály export statikus metódusát használom az eredmények Output ablakra kiküldéséhez.

```
var outString:String=TextConverter.export(flow,TextConverter.TEXT_LAYOUT_FORMAT,
                ConversionType.STRING_TYPE) as String;
trace(outString);
```

Az eredményeket könnyen kimenthetjük egy XML fájlba.

```
var outXML:XML=TextConverter.export(flow,TextConverter.TEXT_LAYOUT_FORMAT,
                ConversionType.XML_TYPE) as XML;
trace(outXML);
```

A szövegformátumnak és XML formátumnak is vannak előnyei és hátrányai egyaránt. Egy XML fájlt könnyebb olvasni és szerkeszteni, de néha nem úgy jeleníti meg a whitespace karaktereket, ahogy akarjuk. Szöveges fájlt nehezebb olvasni és szerkeszteni, viszont a whitespace karakterek megjelenítése pontosabb.

Köszönetnyilvánítás

Ezúton is szeretném megköszönni a Dr. Tornai Róbertnek a diplomamunka elkészítésében nyújtott segítségét, támogatását és tanácsait.