helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

Macromedia Flash MX 2004. Sztuka projektowania

Autorzy: Shane Rebenschied, Lynda Weinman
Tłumaczenie: Marcin Samodulski, Jakub Thiele-Wieczorek
ISBN: 83-7361-806-6
Tytuł oryginału: Macromedia Flash MX 2004 Beyond the Basics Hands-On Training (Hands on Training (H.O.T))
Format: B5, stron: 408

Program Macromedia Flash stał się niemal standardem w zakresie przygotowywania aplikacji multimedialnych na strony WWW, i nie tylko. Jest połączeniem narzędzia do grafiki wektorowej ze środowiskiem programistycznym, co pozwala na tworzenie nie tylko animacji, ale także elementów interaktywnych, pozwalających użytkownikowi między innymi na sterowanie odtwarzaniem oraz wprowadzanie i pobieranie danych. Aby do tworzonej we Flashu prezentacji dodać możliwość komunikacji z użytkownikiem, trzeba zastosować język ActionScript oraz moduły noszące nazwę „komponentów".

Książka „Macromedia Flash MX 2004. Sztuka projektowania" jest przeznaczona dla tych użytkowników Flasha, którzy opanowali już jego narzędzia graficzne i chcą wykonać następny krok – wykorzystać w swoich pracach możliwości oferowane przez ActionScript. Opisuje sposoby używania tego języka do tworzenia pojedynczych elementów i całych aplikacji. Dowiesz się, jak pobierać i formatować dane ze źródeł zewnętrznych, odtwarzać cyfrowe wideo i tworzyć efektowne menu. Dzięki ćwiczeniom i przykładowym projektom zamieszczonym na dołączonym do książki CD-ROM-ie poszerzysz swoje umiejętności o zaawansowane techniki oparte na programowaniu w ActionScript.

- Korzystanie z bibliotek współdzielonych i danych zewnętrznych
- Formatowanie tekstu za pomocą języka HTML i arkuszy stylów
- Tworzenie interaktywnego pokazu slajdów
- Budowanie wskaźnika postępu ładowania danych
- Interaktywne formularze z weryfikacją wprowadzanych danych
- Odtwarzanie plików MP3 i cyfrowego wideo
- Menu stworzone z wykorzystaniem ActionScript
- Publikacja projektu na stronie WWW

Wykorzystaj ActionScript, by tchnąć życie w projekty wykonane we Flashu

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Wstęp **9**

Rozdział 1. Podstawy **19**

O czym jest ta książka? 19
Co powinieneś już umieć? 20
Modularność, modularność, modularność 21
Nowe funkcje programu Flash MX 2004 22
Czy powinieneś dokonać aktualizacji programu Flash MX do MX 2004? 24
Ćwiczenie 1. Co właściwie tworzymy 26
Aktualizacje 30
Gotowe pliki witryny 31

Rozdział 2. Od czego powinienem zacząć? **33**

Na czym polega proces konstruowania witryny WWW? 33
Od jakiego programu należy zacząć? 39
Ćwiczenie 1. Instalacja czcionki używanej w witrynie 40
Ćwiczenie 2. Integracja 41
Sugestie dotyczące pracy nad witryną 53

Rozdział 3. Zaczynamy **55**

Czym jest główny plik SWF? 55
Czym są klasy, obiekty, metody i właściwości? 56
Czym są typy danych? 57
Czym jest funkcja? 58
Czym jest MovieClipLoader i czym różni się od loadMovie? 60
Ćwiczenie 1. Tworzenie głównego pliku SWF i umieszczanie w nim obiektu MovieClipLoader 61
Czym jest biblioteka współdzielona? 72
Ćwiczenie 2. Tworzenie i przygotowywanie biblioteki współdzielonej 72

Rozdział 4. Klasa LoadVars **89**

Ćwiczenie 1. Co właściwie tworzymy 89
Ćwiczenie 2. Wstawianie czcionek współdzielonych 90
Pisanie komentarzy 94

Ćwiczenie 3. Programowanie obiektu LoadVars ... 96
Czym jest klasa LoadVars? ... 104
Ćwiczenie 4. Ładowanie tekstu sekcji „Our History” ... 104
Ćwiczenie 5. Umożliwienie przewijania tekstu ... 111

Rozdział 5. Język HTML i kaskadowe arkusze stylów ... 121
Ćwiczenie 1. Co właściwie tworzymy ... 121
Ćwiczenie 2. Modyfikacja załadowanego tekstu przy użyciu języka HTML ... 122
Kaskadowe arkusze stylów w programie Flash MX 2004 ... 129
Ćwiczenie 3. Korzystanie z kaskadowych arkuszy stylów ... 130

Rozdział 6. Klasa TextFormat ... 141
Ćwiczenie 1. Co właściwie tworzymy ... 141
Czym jest klasa TextFormat ... 143
Ćwiczenie 2. Dodawanie podmenu ... 144
Ćwiczenie 3. Automatyczna zmiana rozmiaru pola tekstowego ... 153
Ćwiczenie 4. Definiowanie stanów Roll Over i Roll Out dla opcji podmenu ... 156
Ćwiczenie 5. Wyłączanie interaktywności ... 160
Ćwiczenie 6. Kopiowanie i wklejanie funkcji ... 162
Ćwiczenie 7. Włączanie interaktywności ... 164
Ćwiczenie 8. Wykańczanie podmenu ... 169
Ćwiczenie 9. Tworzenie podsekcji Our Staff ... 172

Rozdział 7. Tworzenie pokazu slajdów ... 181
Ćwiczenie 1. Co właściwie tworzymy ... 181
Ćwiczenie 2. Przygotowania ... 182
Ćwiczenie 3. Ładowanie pierwszego slajdu ... 187
Ćwiczenie 4. Wyświetlanie liczby dostępnych slajdów ... 192
Ćwiczenie 5. Dodawanie funkcji następny slajd ... 197
Ćwiczenie 6. Dodawanie funkcji poprzedni slajd ... 202
Ćwiczenie 7. Dodawanie tekstu opisującego slajd ... 207
Ćwiczenie 8. Dodawanie licznika slajdów ... 210

Rozdział 8. Tworzenie wskaźnika postępu ładowania ... 217
Ćwiczenie 1. Co właściwie tworzymy ... 217
Ćwiczenie 2. Zacznijmy ... 219
Ćwiczenie 3. Listener onLoadProgress ... 221

Rozdział 9. Tworzenie formularza ... 227
Ćwiczenie 1. Co właściwie tworzymy ... 227
Ćwiczenie 2. Projektowanie formularza ... 230

Ćwiczenie 3. Proste sprawdzanie poprawności formularza ... 239
Ćwiczenie 4. Użycie obiektu LoadVars do wysyłania zawartości formularza ... 248
Ćwiczenie 5. Stylizacja formularza ... 254

Rozdział 10. Odtwarzacz plików MP3 ... 261
Ćwiczenie 1. Co właściwie tworzymy ... 261
Czym różni się pobieranie progresywne od strumieniowania? ... 263
Kompresja strumieniowanego dźwięku ... 264
Ćwiczenie 2. Zaczynamy ... 266
Ćwiczenie 3. Ładowanie pliku MP3 ... 269
Czym jest klasa Sound? ... 274
Ćwiczenie 4. Zatrzymywanie i wznawianie odtwarzania muzyki ... 276
Ćwiczenie 5. Wyświetlanie informacji ID3 ... 285
Ćwiczenie 6. Zmiana ścieżek ... 289

Rozdział 11. Tworzenie odtwarzacza wideo ... 301
Ćwiczenie 1. Co właściwie tworzymy ... 302
Ćwiczenie 2. Zaczynamy ... 303
W jaki sposób utworzyć plik FLV ... 307
Osadzać czy nie osadzać? ... 310
Ćwiczenie 3. Coś tam ... 311
Ćwiczenie 4. Pobieranie i odtwarzanie materiału wideo ... 320
Ćwiczenie 5. Przełączany przycisk Play-Pause ... 330
Czym jest klasa NetStream? ... 333
Ćwiczenie 6. Tworzenie pasku postępu odtwarzania ... 335
Ćwiczenie 7. Użycie funkcji obsługi zdarzenia onStatus ... 341
Ćwiczenie 8. Sprzątanie ... 345

Rozdział 12. Główne menu ... 349
Ćwiczenie 1. Co właściwie tworzymy ... 349
Ćwiczenie 2. Przygotowywanie pliku FLA ... 351
Ćwiczenie 3. Skrypt dla zdarzeń onRollOver, onRollOut i onRelease ... 354
Ćwiczenie 4. Łączenie wszystkich elementów ... 359

Rozdział 13. Przygotowania do zaprezentowania projektu światu ... 365
Ćwiczenie 1. Umieszczanie projektu opracowanego we Flashu na stronie napisanej w języku HTML ... 365
Potencjalne zmiany w sposobie osadzania elementów na stronach WWW ... 373
Opcje dotyczące wykrywacza pluginu ... 374
Ćwiczenie 2. Tworzenie detektora pluginu Flash MX 2004 ... 377
Podsumowanie ... 383

Dodatek A Obsługa techniczna i rozwiązywanie problemów **385**

Dodatek B Zasoby związane z programem Macromedia Flash MX 2004 ..... **389**

Dodatek C Flash Communication Server i skrypty CGI **395**

Skorowidz **397**

3 Zaczynamy

W tym rozdziale:

- Główny plik SWF
- Klasy, obiekty, metody i właściwości
- Typy danych
- Funkcje
- Klasa MovieClipLoader
- Biblioteki współdzielone
- Tworzenie i przygotowywanie biblioteki współdzielonej

W poprzednich dwóch rozdziałach poznałeś podstawowe koncepcje związane z tworzeniem witryn WWW w programie Flash MX 2004. Teraz wykorzystamy je w praktyce. W tym rozdziale utworzysz główny plik SWF, który będzie pełnił rolę pojemnika przechowującego inne pliki związane z witryną oraz kod w języku ActionScript możliwy do wielokrotnego wykorzystania. Napiszesz również główny skrypt, kontrolujący sposób ładowania wszystkich plików SWF i JPEG używanych w witrynie. Dodatkowo dowiesz się, czym są biblioteki współdzielone, pozwalające na przechowywanie zasobów (takich jak czcionki) wykorzystywanych w projekcie, a także stworzysz jedną z nich. Jest to dosyć długi rozdział zajmujący się wieloma nowymi i bardzo ważnymi tematami, więc jeśli czytasz to późną nocą i myślisz sobie „tylko dokończę ten rozdział i idę spać", radzę Ci poczekać z lekturą do czasu, gdy się porządnie wyśpisz. Chyba że chcesz mieć naprawdę dziwne sny, w których będą Cię goniły wielkie `MovieClipLoadery` z wielkimi, kłapiącymi kłami. ;-)

Czym jest główny plik SWF?

Jeśli nie mieszkasz w jaskini od roku 1954, zapewne słyszałeś o książkach J.R.R. Tolkiena, przeniesionych na ekran w postaci trylogii *Władca pierścieni*. Dzisiejsze wszędobylskie media stamtąd podchwyciły zdanie „Jeden by wszystkimi rządzić", związane z magicznym, wszechmocnym pierścieniem, którego moc pozwala na kontrolowanie kilku innych magicznych pierścieni. W poniższym ćwiczeniu stworzymy magiczny pierścień naszej witryny. Ten główny plik SWF spełni rolę pojemnika, do którego będą ładowane wszystkie inne pliki SWF. Będzie on również zawierał kod w języku ActionScript, który zostanie wielokrotnie wykorzystany w witrynie, na przykład skrypt pozwalający

na ładowanie zasobów (`MovieClipLoader`), zmiennych (`LoadVars`) i kaskadowych arkuszy stylów (*TextField.StyleSheet*). Tworząc go i używając do przechowywania zmiennych i często używanych funkcji, uzyskujemy dwie rzeczy:

- Centralizujemy często używane fragmenty kodu w języku ActionScript. Pozwala to na łatwą modyfikację kodu w jednym, scentralizowanym miejscu. Ponieważ inne pliki SWF odnoszą się do niego, automatycznie będą korzystać ze zmian wprowadzonych w skrypcie. Gdybyś zamiast tego wpisał ten kod bezpośrednio do każdego pliku, który z niego korzysta, to modyfikując plik lub poprawiając błąd, musiałbyś powtórzyć te czynności wielokrotnie, w każdym z używanych plików. Zobaczysz, że centralizacja skryptów znacznie usprawnia proces tworzenia witryny.
- Centralizacja kodu w języku ActionScript pozwala na zmniejszenie rozmiaru plików, ponieważ wszystkie skrypty są trzymane tylko w jednym miejscu. Gdybyś wstawił je do każdego pliku SWF, znacznie powiększyłbyś całkowity rozmiar projektu. Trzymanie często używanego kodu w jednym miejscu i korzystanie z referencji w innych plikach SWF pozwala na redukcję rozmiaru plików poprzez uniknięcie powtarzania identycznego skryptu.

Czym są klasy, obiekty, metody i właściwości?

Gdy zaczniesz pisać kod w języku ActionScript, który nada funkcjonalność tej witrynie, napotkasz pewne słowa, których prawdopodobnie nie znasz. Ta sekcja w skrócie opisuje definicje klas, obiektów, metod i właściwości. Wierzę w to, że analogie pomagają w procesie uczenia. Aby pomóc Ci w lepszym zrozumieniu tych skomplikowanych tematów, posłużę się w swoich wyjaśnieniach psem. Choć część terminologii może brzmieć nieco mylnie, gdy dokończysz ćwiczenia zawarte w tej książce, będziesz już z nią bardziej obeznany.

- **Klasy**. Klasy są definicjami grup obiektów. Na przykład pies może należeć do klasy zwierząt. Wszystkie zwierzęta mają pewne charakterystyczne cechy, które posiada też pies. Klasę można uważać po prostu za kategorię obiektów. W przypadku programu Flash MX 2004 klipy filmowe i przyciski są klasami. Posiadają one metody i właściwości, które definiują, co mogą one robić w projekcie, a czego nie mogą.
- **Obiekty**. Obiekt jest w skrócie instancją klasy. Tak jak możesz posiadać na scenie instancję symbolu graficznego, obiekt w przypadku języka ActionScript jest instancją klasy. Jest on zbiorem właściwości i metod (o których więcej przeczytasz poniżej), dziedziczonych z klasy. Ponadto każdy z obiektów posiada własną nazwę, której można używać w referencjach do niego. Na przykład w programie Flash MX 2004 instancje klipów filmowych i przycisków są obiektami, ponieważ stanowią instancje odpowiadających im klas.
- **Metody**. Metody są zachowaniami obiektów. Zachowaniami (metodami) psa mogą być: szczekanie, sapanie, ślinienie się, sikanie na meble i zjadanie butów. W przypadku języka Flash MX 2004 ActionScript klasy posiadają wbudowane metody, które możesz wykorzystać. Możesz też stworzyć własne metody, przyporządkowując je funkcji (więcej na temat funkcji dowiesz się w dalszej części tego rozdziału). Analogią tej możliwości może być na przykład to, że pies zawsze szczeka (jest to jego wbudowana metoda), ale jeśli chcesz, możesz mu dodać zachowanie, które pozwoli na przemieszczanie kotów siłą woli i rzucanie nimi po pokoju (metoda dodana). Na przykład klip filmowy, będący obiektem, posiada metody pozwalające na jego odtwarzanie, zatrzymywanie odtwarzania, przechodzenie do następnej klatki itd.

Uwaga

Czym są zmienne?

W tej książce często będziemy używać zmiennych, dlatego warto wiedzieć, czym są. W skrócie zmienna jest pojemnikiem tworzonym w języku ActionScript, pozwalającym na przechowywanie danych lub referencji do instancji obiektu lub przyporządkowywanie utworzonych obiektów. Zmienna jest nieco podobna do symbolu. Jednak tak jak symbol jest pojemnikiem pozwalającym na przechowywanie elementów wizualnych — grafiki, tekstu itp., zmienna pozwala na przechowywanie danych, takich jak imię, liczba itp. Dzięki tej książce zyskasz duże doświadczenie w pracy ze zmiennymi. Będziemy ich używać do przechowywania liczb (takich jak czas trwania ścieżki dźwiękowej), ciągów znaków (takich jak imię użytkownika wysyłającego komentarz za pomocą formularza) i obiektów języka ActionScript (takich jak obiekt `MovieClipLoader`*, który utworzymy w dalszej części tego rozdziału).*

Zmienne są wręcz nieocenione w pracy w programie Flash MX 2004

- **Właściwości**. Właściwości są atrybutami, które definiują obiekt. Na przykład wszystkie psy (a przynajmniej większość z nich) posiadają cztery łapy, dwoje oczu, ogon, pysk i śmierdzący oddech. Klip filmowy posiada właściwości takie jak: jego widoczność (`_visible`), przezroczystość (`_alpha`), obrót (`_rotation`) itd.

Z kilkoma z tych określeń będziesz miał do czynienia w ćwiczeniu 1. w tym rozdziale.

Czym są typy danych?

W czasie pisania kodu w języku ActionScript zauważysz zapewne coś, z czym do tej pory się nie spotkałeś. Pisząc akcje i tworząc zmienne, będziesz dodawał krótki tekst po ich nazwie. Jest on nowością we Flashu MX 2004, nazwaną **określeniem typu danych**. Pozwala on na określenie rodzaju danych, jakie będzie zawierać zmienna. Początkowo wydaje się to nieco zagmatwane, ale szybko nabierzesz doświadczenia i zrozumiesz, o czym piszę, gdy zaczniesz korzystać ze zmiennych w ćwiczeniach zawartych w tym i w następnych rozdziałach. Oto przykład określenia typu danych:

```
var myVar:String = "Here is a string of text.";
var myVar2:Number = 1234;
```

Widzisz? Po nazwie zmiennej `myVar` jest napis `:String`, po `myVar2` zaś — `:Number`. Zaznaczone identyfikatory są przykładami określeń typów danych. Żeby użyć ich w swoich akcjach, wystarczy je wpisać po nazwie zmiennej. Zadaniem powyższych określeń jest poinformowanie skryptu, że zmienna `myVar` zawiera dane typu `String` (ciąg znaków), `myVar2` zaś — `Number` (liczba). Takie rozwiązanie ma wiele zalet. Jako średnio zaawansowany użytkownik programu Flash powinieneś zdawać sobie sprawę z trzech najważniejszych:

- Określanie typów pozwala na poinformowanie skryptu, jakiego rodzaju dane ma przechowywać zmienna. Pozwala również na zmianę typu danych przechowywanych w zmiennej. Innymi słowy, jeśli zawiera ona liczbę, która ma być traktowana jako ciąg znaków, możesz to zapewnić używając typu `String`. Dlaczego miałbyś to zrobić?

Cóż, jeśli wykonujesz jakieś obliczenia (dzielenie, mnożenie itd.) na liczbach, chciałbyś, żeby były one traktowane jako typ `Number`, a nie `String`. Próba wykonania obliczeń matematycznych (z którymi będziesz miał do czynienia w czasie tworzenia paska postępu i preloaderów) na ciągach znaków powoduje powstanie błędów.

- Korzystanie ze zmiennych i obiektów określonego typu jest uważane za dobry styl kodowania. A to dlatego, że określa się w ten sposób typ danych przechowywanych w zmiennej lub obiekcie (`String`, `Number` itp.). Dostarczenie takiej informacji Flashowi pozwala na zmniejszenie potencjalnej liczby błędów w filmie.
- Zauważ również, że ze względu na to, iż w programie Flash MX 2004 został usunięty tryb *Normal* w panelu *Actions* (akcje), określenie typu zmiennych pomaga w szybszym pisaniu skryptów. Dzieje się tak dlatego, że dzięki niemu wyświetlane są podpowiedzi do kodu i możliwe jest korzystanie z funkcji automatycznego uzupełniania w przypadku wbudowanych *klas* (*obiektów*).

Powinienem Cię poinformować o tym, że określanie typów w języku ActionScript nie jest obowiązkowe. Dlatego też jest obsługiwane tylko w czasie tworzenia pliku SWF z użyciem języka ActionScript 2.0 (czyli wersji używanej domyślnie w momencie tworzenia nowego pliku FLA w programie Flash MX 2004). Jeśli pominiesz określanie typów, w większości przypadków akcje będą działały bez większych problemów. Chciałem o tym napisać już na początku książki, ponieważ jest to nowa cecha języka ActionScript 2.0. Ponadto moim zadaniem jest nauczenie Cię właściwych sposobów pisania skryptów, a także przygotowanie do nauki bardziej zaawansowanych funkcji tego języka, poprzez zaprezentowanie niektórych sformułowań i technik. W dalszej części tego rozdziału dowiesz się, w jaki sposób możesz tworzyć własne określenia typów danych.

Czym jest funkcja?

Symbol można uważać za pojemnik, w którym przechowywana jest grafika. Jego ciekawą cechą jest to, że można go wykorzystywać wiele razy bez znacznego zwiększenia rozmiaru pliku SWF. Funkcje są pod tym względem podobne, jednak zamiast grafiki przechowują fragmenty kodu w języku ActionScript. Można je uważać za pojemniki pozwalające na przechowywanie skryptów, które mogą zostać wielokrotnie użyte w projekcie. Są wyjątkowo użyteczne w sytuacjach, w których występuje potrzeba powtarzania bloku instrukcji.

Powiedzmy, że posiadamy szereg przycisków, jak na pasku nawigacyjnym naszej witryny. Chcemy, żeby w momencie, gdy użytkownik kliknie jeden z nich, pojawił się animowany klip filmowy, muzyka ucichła, a w dynamicznym polu tekstowym, umieszczonym na scenie, pojawił się jakiś tekst. Gdybyśmy nie użyli funkcji, musielibyśmy przypisać wszystkie te akcje każdej instancji przycisku. Następnie aby je zmienić, powinniśmy modyfikować kod każdej z nich! Co, jeśli przycisków byłoby 50? To prawdziwy przepis na szaleństwo. Zamiast tego możemy utworzyć funkcję zawierającą kod, który wykonuje te same trzy akcje (rozpoczęcie odtwarzania klipu, zatrzymanie muzyki i wyświetlenie komunikatu), a następnie przyporządkować instancjom przycisków po jednym wierszu kodu, powodującym wywołanie funkcji i, co za tym idzie, wykonanie zawartych w niej akcji. Wówczas, gdy zaistnieje potrzeba modyfikacji kodu, nie będziemy musieli go zmieniać w każdej z instancji. Wystarczy, że zmodyfikujemy funkcję. Oto przykład tego, jak może ona wyglądać:

```
function buttonClick() {
   myMC.play();
   myMusic.stop();
   myMessage.text = "You clicked on a button. Congratulations!";
}
```

W tym przykładzie funkcji została nadana nazwa `buttonClick`. Nazwa może być zupełnie dowolna. Jedynym ograniczeniem jest to, że nie może zaczynać się od cyfry, zawierać spacji ani znaków specjalnych (?, *, % itp.). Pomiędzy nawiasami klamrowymi (`{ }`) znajdują się akcje, które mają zostać wykonane w momencie wywołania funkcji. Aby to nastąpiło, wystarczy wpisać jej nazwę (upewniając się, że określona została ścieżka dostępu do niej w przypadku, gdy znajduje się ona na innej listwie czasowej), dodając operator wywołania `()`:

```
buttonClick();
```

Aha, nowością w programie Flash MX 2004 jest to, że rozróżnia on małe i wielkie litery w skryptach. Tak więc jeśli napiszesz:

```
buttonclick();
```

funkcja nie zostanie wywołana.

Inną zaletą korzystania z funkcji jest możliwość przekazywania im parametrów. Oznacza to, że możesz przekazywać funkcji parametry, niezależnie od tego, gdzie została ona wywołana (w powyższym przykładzie jest wywoływana przez przycisk). Na przykład, co zrobić, jeśli chcemy, żeby treść wyświetlanego komunikatu zależała od tego, który przycisk został kliknięty? W tej chwili komunikat jest stałym elementem funkcji, co oznacza, że niezależnie od tego, który przycisk kliknie użytkownik, wyświetlony zostanie ten sam komunikat. Możemy jednak przyporządkować funkcji parametr, następnie przy wywoływaniu funkcji w akcjach poszczególnych przycisków przekazać do tego parametru inny tekst. Jeśli wydaje Ci się, że piszę po grecku, może pomoże następujący przykład. Ta sama funkcja z dodanym parametrem będzie wyglądała następująco:

```
function buttonClick (msg) {
   myMC.play();
   myMusic.stop();
   myMessage.text = msg;
}
```

Jak widzisz, jedyną różnicą jest to, że obecnie po nazwie funkcji w nawiasach został umieszczony napis `msg`. Nie, nie oznacza on dodatku do żywności, poprawiającego jej smak. Jest to po prostu nazwa zmiennej zawierającej tekst, który będzie wyświetlany, będąca skrótem od `message` (ang. *komunikat*). Jeśli przyjrzysz się ostatniej akcji, zauważysz, że znajduje się ona również po instrukcji `myMessage.text =`. Parametr `msg` działa po prostu jak miejsce na informację, którą możemy przekazać funkcji w momencie jej wywołania. Na przykład jeśli chcesz, żeby po kliknięciu przycisku o nazwie *Our Products* funkcja wyświetlała komunikat *„We have lots of stuf. Check it out.”*, powinieneś przyporządkować mu następującą akcję:

```
on (release) {
   buttonClick ("We have lots of stuff. Check it out.");
}
```

Tekst *„We have lots of stuff. Check it out."* jest parametrem przekazywanym funkcji w momencie, gdy użytkownik kliknie przycisk. Zostaje on umieszczony w miejscu, w którym znajduje się napis `msg`. Dzięki temu w momencie kliknięcia tego przycisku w dynamicznym polu tekstowym zostaje wyświetlony komunikat zawierający ten ciąg znaków.

Parametr funkcji jest po prostu zmienną lokalną. To znaczy, że po wywołaniu funkcji zostaje on usunięty dla zaoszczędzenia pamięci. Istnieje nawet możliwość tworzenia zmiennych o nazwie takiej samej jak nazwa parametru. Nie powoduje to żadnych konfliktów.

Funkcję użytą w poprzednich przykładach należy stworzyć za pomocą języka ActionScript. Nazywamy ją funkcją najwyższego poziomu. Obiekty również posiadają wbudowane funkcje (co sam zobaczysz, gdy będziesz tworzył obiekt `MovieClipLoader` w dalszej części tego rozdziału). Są one nazywane metodami i zostały opisane wcześniej w części „Czym są klasy, obiekty, metody i właściwości".

Jestem pewien, że nadal masz wiele pytań związanych z funkcjami i sposobami ich wykorzystania, nauczysz się o nich wiele z dalszej części tej książki.

Czym jest MovieClipLoader i czym różni się od loadMovie?

Funkcja `loadMovie` pozwala na załadowanie pliku SWF lub JPG do innego pliku SWF. Jest to świetne rozwiązanie, ponieważ pozwala na podzielenie dużego projektu na mniejsze części i umożliwia ładowanie ich za każdym razem, gdy użytkownik tego wymaga. Jest to również doskonały sposób na skrócenie czasu ładowania witryn opartych na programie Flash MX 2004, ponieważ daje możliwość początkowego załadowania jedynie podstawowej informacji (takiej jak menu nawigacyjne). Następnie w miarę zapotrzebowania ściągane są inne sekcje witryny.

Funkcji `loadMovie`, mimo że była bardzo użyteczna, brakowało pewnych możliwości, dlatego wielu programistom nie wystarczała. Potrzebowali lepszych procedur, pozwalających na konstruowanie preloaderów, zarządzanie błędami itp. Dlatego firma Macromedia zdecydowała się wprowadzić w programie Flash MX 2004 nowy (lepszy) sposób ładowania zewnętrznych zasobów. Tak powstała klasa `MovieClipLoader`.

Klasa `MovieClipLoader` zajmuje się tym samym, co funkcja `loadMovie` — pozwala na ładowanie zewnętrznych plików SWF i JPG do innego pliku SWF (lub innego poziomu). Tym, co ją odróżnia, jest fakt, że jest znacznie bardziej złożona i oferuje więcej opcji związanych z ładowaniem i zajmowaniem się plikami SWF i JPG.

`MovieClipLoader` jest klasą, której należy przyporządkować obiekty nasłuchujące, czyli **listenery**. Jak wskazuje nazwa, „nasłuchują" one, czym w danym momencie zajmuje się `MovieClipLoader`. Gdy wystąpi określone zdarzenie, wykonają powierzone im zadania. Na przykład jeśli używasz obiektu `MovieClipLoader` w celu załadowania zewnętrznego pliku JPG, a plik ten nie istnieje, możesz przyporządkować mu `listener` `onLoadError`, który spowoduje wyświetlenie komunikatu o błędzie. Kiedyś, gdy do ładowania zasobów używało się funkcji `loadMovie`, jeśli chciało się stworzyć preloader pokazujący stopień załadowania pliku, należało stworzyć pętlę, która powtarzała kod w języku

ActionScript. Teraz klasa `MovieClipLoader` zawiera własne pętle tego typu, w obiekcie listener o nazwie `onLoadProgress`. Pod wieloma względami nowa klasa jest znacznie łatwiejsza w obsłudze, a także oddaje do dyspozycji znacznie większą liczbę opcji pozwalających na obsługę zewnętrznych plików SWF i JPG. W tabeli poniżej zostały przedstawione różne funkcje typu listener wraz z opisami.

Listener	Opis
`onLoadComplete`	Wywoływany jest (a co za tym idzie, są wykonywane wszystkie zawarte w nim akcje) w momencie, gdy zakończy się ładowanie danego zasobu.
`onLoadError`	Wywoływany jest, gdy ładowanie zasobu się nie powiedzie.
`onLoadInit`	Podobnie jak `onLoadComplete` wywoływany jest w momencie, gdy zakończy się ładowanie danego zasobu i gdy uruchomiona zostaje pierwsza klatka załadowanego klipu. W niektórych przypadkach lepiej jest stosować go zamiast `onLoadComplete`, ponieważ w momencie jego wywołania wiadomo, że klip został do końca załadowany i jest gotowy do odtworzenia.
`onLoadProgress`	Wywoływany za każdym razem, gdy na dysku jest zapisywana ładowana zawartość klipu. Oznacza to, że za każdym razem, gdy zostaje ściągnięty fragment informacji dotyczącej klipu, jest on uruchamiany. Jest on pętlą w języku ActionScript, która jest wykonywana w czasie ściągania zasobów. Właśnie w nim umieszcza się kod preloadera (zrobimy to w rozdziale 8. „Tworzenie preloadera”).
`onLoadStart`	Wywoływany w momencie, gdy rozpoczyna się ładowanie zasobu.

Jak widzisz (ale zapewne jeszcze nie do końca rozumiesz), klasa `MovieClipLoader` jest bardzo potężnym dodatkiem do programu Flash MX 2004. W dalszej części tego rozdziału wykorzystamy ją po raz pierwszy.

Ćwiczenie 1. Tworzenie głównego pliku SWF i umieszczanie w nim obiektu MovieClipLoader

W tym ćwiczeniu zaczniemy tworzyć pierwszy fragment witryny firmy L.A. Eyeworks, czyli główny plik SWF. Jak wiesz z części „Czym jest główny plik SWF?”, plik *master.swf* będzie służył za pojemnik, do którego będą ładowane inne moduły w formacie SWF (muzyka, oprawki, sekcja *about us* itp.). Będą w nim również przechowywane często wykorzystywane fragmenty kodu w języku ActionScript. Przechowywanie często używanych akcji w centralnej lokalizacji pozwala na skrócenie czasu spędzonego nad projektem, ponieważ dzięki niemu nie ma potrzeby powtarzania kodu za każdym razem, gdy jest on potrzebny. Dużo łatwiej jest również wprowadzać późniejsze zmiany w skrypcie. W tym ćwiczeniu napiszemy skrypt `MovieClipLoader`, który zajmie się ładowaniem plików SWF i JPG do witryny.

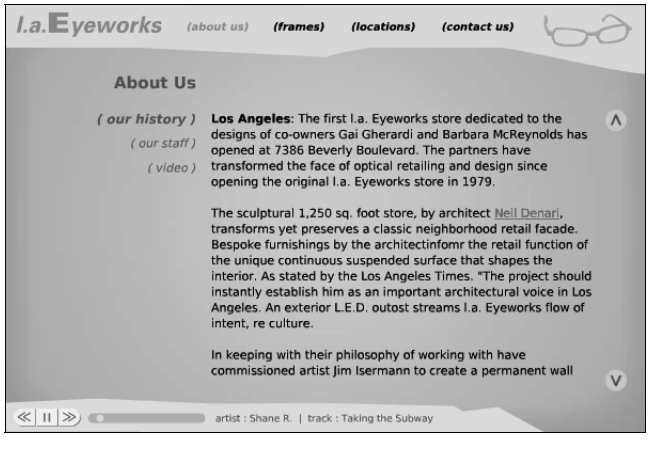
W tym ćwiczeniu nauczysz się wielu rzeczy, więc jeśli chcesz najpierw zrobić sobie przerwę lub przebiec się dookoła bloku, teraz jest na to odpowiedni czas. ;-)

1. Otwórz swoją ulubioną przeglądarkę i wpisz do niej adres:

   http://www.lynda.com/flashbtb/laeyeworks/

Spowoduje to otworzenie w jej oknie gotowej witryny. W zależności od szybkości łącza jej ładowanie może trwać bardzo krótko lub nawet minutę. Po załadowaniu witryny zobaczysz pasek nawigacyjny na górze ekranu, odtwarzacz muzyczny na dole, na środku zaś animowaną grafikę firmową. Każdy z tych elementów jest osobnym plikiem SWF, odpowiednio *main_menu.swf*, *music.swf* i *splash.swf*. Wszystkie trzy pliki są w momencie załadowania witryny automatycznie ładowane do głównego pliku *master.swf*.

Gdy klikniesz jeden ze znajdujących się na górze przycisków, na przykład *about us*, zobaczysz, że grafika zniknie, a na jej miejsce zostanie załadowany moduł *About Us* (inny plik SWF). To samo dotyczy dowolnego innego przycisku nawigacyjnego. Środek witryny jest miejscem, do którego są ładowane różne moduły, podmieniane w momencie kliknięcia jednego z przycisków. Cała zawartość jest ładowana i przechowywana w pliku *master.swf*, który za chwilę utworzymy. Gdy użytkownik klika jeden z przycisków, każe instancji klasy `MovieClipLoader` załadować określony plik SWF lub JPG. Czy to wydaje się zbyt łatwe? W gruncie rzeczy takie właśnie jest. Oczywiście żeby do tego doprowadzić, potrzeba odrobiny pracy i przemyślanego kodu w języku ActionScript, ale przecież ta książka istnieje po to, żeby Cię tego wszystkiego nauczyć, nieprawdaż? :-)



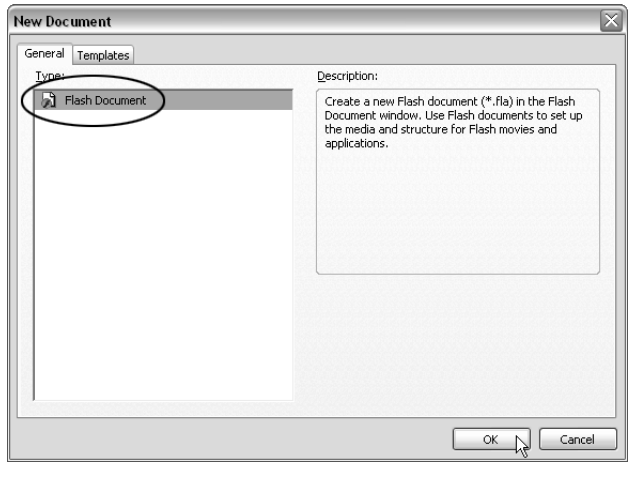
2. Na dołączonej do książki płycie CD-ROM znajduje się folder o nazwie *la_eyeworks*. Zawiera on wszystkie pliki, z którymi będziemy pracować w tej książce. Skopiuj go na pulpit wraz z zawartością.

Często będziemy korzystać z plików znajdujących się w tym folderze. Dlatego upewnij się, że dokładnie wiesz, gdzie się on znajduje. Folder ten będzie również pełnił funkcję głównego katalogu witryny L.A. Eyeworks. Innymi słowy, będziesz w nim zapisywał wszystkie pliki, którymi będziemy się zajmować.

Uwaga

Jeśli pracujesz na komputerze działającym pod kontrolą systemu operacyjnego Windows, wykonaj instrukcje przedstawione we wstępie, aby umożliwić odczyt i zapis folderu i plików, które się w nim znajdują.

3. Uruchom program Flash MX 2004. Wybierz z menu polecenie *File/New* (plik/nowy), a następnie w oknie dialogowym *New Document* — opcję *Flash Document* i kliknij *OK*.

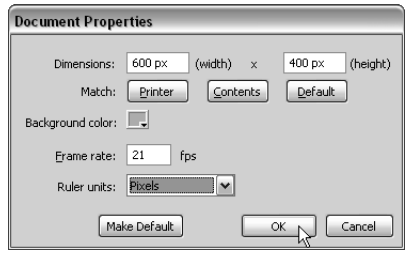


Uwaga

Niektóre okna w programie Flash MX 2004 wyglądają inaczej niż w wersji Flash MX Professional 2004. Przykładem jest strona startowa. Okno dialogowe New Document (pokazane na powyższej ilustracji) również do nich należy. W programie Flash MX Professional 2004, w zakładkach General i Templates znajduje się więcej opcji. Pomimo tych różnic ćwiczenia zawarte w książce dotyczą obu wersji programu.

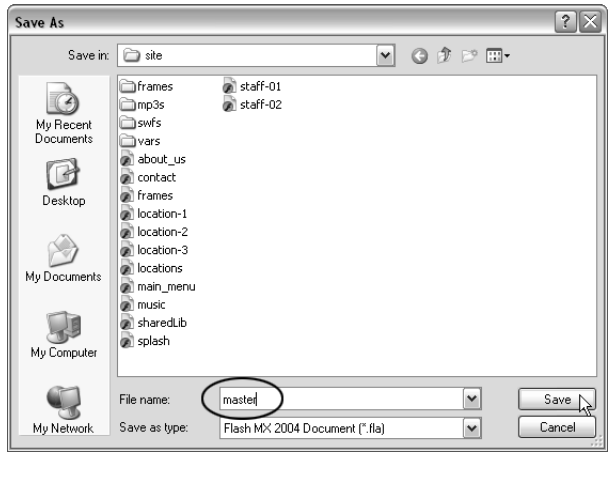
Gdy plik SWF zostaje załadowany do innego pliku tego typu, plik nadrzędny definiuje prędkość odtwarzania filmu. Oznacza to, że plik *master.swf* (który właśnie utworzyłeś) określa ten parametr dla wszystkich ładowanych do niego plików SWF. W przypadku witryny firmy L.A. Eyeworks został on ustawiony na 21 klatek na sekundę. Choć jego wartość jest przedmiotem sporu pomiędzy różnymi projektantami korzystającymi z programu Flash MX 2004, moim zdaniem ta, której użyliśmy, wystarczy, żeby wszystkie animacje były płynnie odtwarzane (niższa prędkość odtwarzania filmu powoduje, że wyświetlanie obrazu nie jest płynne). Nie jest również zbyt wymagająca, jeśli chodzi o parametry komputerów użytkowników, którzy będą oglądali witrynę (większa prędkość odtwarzania wymaga lepszego sprzętu).

4. Otwórz okno *Document Properties*, wciskając *Ctrl+J* (w systemie Windows) lub *Cmd+J* (w systemie Mac OS). W polach *Dimensions* ustaw szerokość filmu na 600 pikseli, a wysokość na 400 pikseli. Ustaw kolor tła na barwę o wartości szesnastkowej #B4CCE5. Możesz to zrobić klikając kolorowy kwadrat i wpisując ją w polu znajdującym się na prawo od okienka (wartość została określona na podstawie modelu witryny stworzonego w programie Adobe Photoshop). W polu *Frame rate* wpisz prędkość odtwarzania filmu równą 21 klatek na sekundę. Kliknij *OK*.



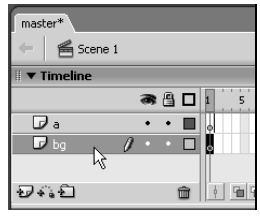
Wskazówka *Jeśli klikniesz przycisk Make Default, program użyje wpisanych ustawień jako wartości domyślnych dla wszystkich tworzonych dokumentów.*

5. Wybierz z menu polecenie *File/Save* (plik/zapisz) i zapisz plik FLA. W oknie dialogowym przejdź do pulpitu, następnie wejdź do folderu *la_eyeworks*, który umieściłeś tam na początku ćwiczenia, i do katalogu *site*. Nadaj plikowi nazwę `master` i kliknij *Save*.

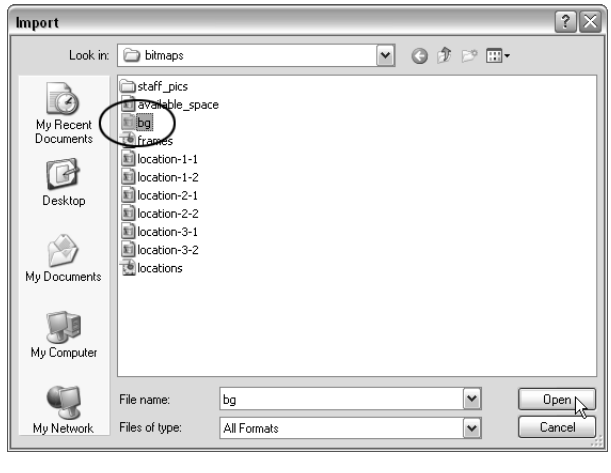


Kolejnym wspólnym elementem wszystkich ładowanych plików SWF jest tło. Niezależnie od tego, który plik obecnie ogląda użytkownik, w tle znajduje się ten sam obraz. W następnym kroku dodamy go do pliku *master.fla*.

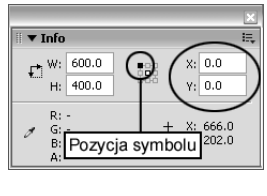
6. Zmień nazwę warstwy 1. na `bg`, po czym utwórz nową warstwę i zmień jej nazwę na `a`. Umieścimy na niej kod w języku ActionScript. Warstwa `bg` będzie zaś zawierać (tak, zgadłeś) obraz wyświetlany w tle. Zanim przejdziemy do następnego kroku, upewnij się, że zaznaczyłeś warstwę `bg`.



7. Wciśnij *Ctrl+R* (w systemie Windows) lub *Cmd+R* (w systemie Mac OS), aby zaimportować na scenę nowy element. W oknie dialogowym *Import* przejdź do pulpitu, następnie wejdź do folderu *la_eyeworks*, *resources*, a następnie *bitmaps*. Zaznacz bitmapę o nazwie `bg` i kliknij przycisk *Open* (w systemie Windows) lub *Import* (w systemie Mac OS), aby umieścić ją na scenie.



Bitmapa `bg` (która posiada taki sam rozmiar jak scena — 600×400 pikseli) powinna zostać automatycznie wyrównana z prawym górnym rogiem sceny. Możesz to sprawdzić: otwórz panel *Info* (*Ctrl+I* w systemie Windows lub *Cmd+I* w systemie Mac OS) i upewniaj się, że punkt, względem którego jest ustalana pozycja symbolu jest umieszczony w prawym górnym rogu, a jego obie współrzędne mają wartość 0.

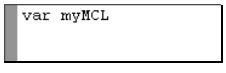


8. Zablokuj warstwę `bg` i kliknij pierwszą klatkę warstwy `a` (akcje). Otwórz panel *Actions* (*F9*).

   Zanim będziesz mógł wykorzystać klasę `MovieClipLoader`, musisz utworzyć jej instancję za pomocą języka ActionScript. Przypomina to sposób użycia obrazka — najpierw należy go narysować, przekonwertować do postaci symbolu, a następnie umieścić na scenie. Klasa `MovieClipLoader` musi być użyta w podobny sposób. Najpierw należy utworzyć jej instancję (nazywaną obiektem). Zajmiemy się tym w następnym kroku.

9. W panelu *Actions* wpisz następujący wiersz kodu:

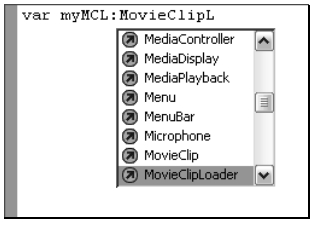
```
var myMCL
```



   Jest to instrukcja, która każe programowi Flash MX 2004 utworzyć zmienną o nazwie `myMCL`. Tekst *var* jest skrótem od *variable* (zmienna). Pozwala on również na określenie zakresu zmiennej (ang. *variable scope*), którym zajmiemy się w dalszej części książki. Tak więc jeśli zobaczysz przed jakąś nazwą słowo *var* (w tym przypadku przed ciągiem znaków `myMCL`), oznacza to, że tworzona jest zmienna o podanej nazwie. Choć jego użycie nie zawsze jest konieczne, korzystanie z niego jest uważane za dobry nawyk. Jest ono również konieczne w przypadku przyporządkowywania zmiennej określonego typu (czym zajmiemy się za chwilę). W następnym kroku przyporządkujemy zmiennej obiekt klasy `MovieClipLoader`. W ten sposób, gdy będziemy chcieli wydać mu jakiś rozkaz, będziemy się do niego odnosić używając nazwy zmiennej, `myMCL`.

10. Po deklaracji zmiennej `myMCL` wpisz:

```
:MovieClipLoader
```



   Gdy będziesz wpisywał powyższy ciąg znaków, zauważysz, że pojawi się przewijane pole tekstowe, zawierające różne opcje. Jeśli nie chcesz go wpisywać, możesz wybrać tę opcję z listy i wcisnąć *Enter* lub *Return*. Flash MX 2004 wstawi wówczas to słowo. Pokazana lista jest przykładem podpowiedzi do kodu, czyli próby pomocy ze strony programu. Na podstawie tego, co piszesz, Flash MX 2004 wykrywa, że używasz deklaracji typu (określonej za pomocą dwukropka) pozwalającej na ustawienie typu zmiennej `myMCL`. W tym przypadku informujesz go, że typem ma być `MovieClipLoader`. Gdybyś pominął tę deklarację, skrypt nadal by działał. Jednak po wprowadzeniu do programu Flash MX 2004 języka ActionScript 2.0 zaleca się stosowanie dokładnych określeń typów, ponieważ jest to zgodne z jego specyfikacją. Innymi słowy, korzystanie z określeń typów jest zalecane przez firmę Macromedia, ponieważ uznaje się je za dobry nawyk programistyczny. Głównie dlatego będziemy z nich korzystać w ćwiczeniach.

11. Dopisz na końcu wiersza:

```
= new MovieClipLoader();
```

```
var myMCL:MovieClipLoader = new MovieClipLoader();
```

W ten sposób każesz Flashowi stworzyć nową instancję obiektu `MovieClipLoader` i umieścić ją w zmiennej `myMCL`. W przyszłości, jeśli będziesz chciał, aby Flash wykonał jakąś czynność, wystarczy, że odniesiesz się do tego obiektu, używając nazwy zmiennej.

Gratulacje! Właśnie utworzyłeś nowy obiekt typu `MovieClipLoader`, który wkrótce będzie mógł być używany do ładowania plików SWF i JPG do głównej zawartości strony. Nawet o tym nie wiedząc, dodałeś projektowi sporo nowej funkcjonalności. Podobnie jak umieszczenie w projekcie klipu automatycznie daje możliwość wykorzystania języka ActionScript do modyfikacji jego metod i właściwości, takich jak widoczność, przezroczystość, obrót itp., dodanie obiektu `MovieClipLoader` daje możliwość korzystania z jego wbudowanych metod i nasłuchiwaczy (listenerów) do ładowania plików SWF i JPG. Umożliwia to również uzyskiwanie raportów pokazujących postępy w ściąganiu itp. Dodanie tego jednego wiersza kodu dało więcej korzyści, niż sobie wyobrażasz. Poznasz je w kolejnych ćwiczeniach.

Teraz, choć stworzyłeś nowy obiekt typu `MobieClipLoader`, nadal potrzebujesz kilku wierszy kodu w języku ActionScript, aby zaczął on działać. Gdy będzie on gotowy, będziesz miał możliwość wykorzystania go do wielu czynności, takich jak ładowanie pliku SWF. Jednak jeśli chcesz dowiedzieć się więcej na temat procesu ładowania, np. jaki fragment pliku pozostał jeszcze niezaładowany lub co należy zrobić po jego załadowaniu itp., nie pytaj o to tego obiektu. Aby uzyskać informacje na temat tego, co aktualnie robi `MovieClipLoader`, musisz mu przyporządkować obiekt nasłuchujący (listener). Odpowiada on na różne zdarzenia wywoływane przez tę klasę. Na przykład jeśli każesz obiektowi typu `MovieClipLoader` załadować plik SWF, zacznie on to robić, a gdy skończy, wyśle parametr (`success`) do obiektu nasłuchującego (listenera) o nazwie `onLoadComplete` (jeśli taki dodałeś), przyporządkowanego do obiektu `MovieClipObject`, do którego jest ładowany plik. Obiektowi nasłuchującemu (listenerowi) możesz przypisać skrypt, który określi, co ma się zdarzyć po kompletnym załadowaniu pliku.

12. Kliknij na końcu pierwszego wiersza kodu i wciśnij klawisz *Enter* (w systemie Windows) lub *Return* (w systemie Mac OS), aby utworzyć nowy. Następnie napisz:

```
var myListener:Object = new Object();
```

```
var myMCL:MovieClipLoader = new MovieClipLoader();
var myListener:Object = new Object();
```

Ta instrukcja każe Flashowi MX 2004 utworzyć nową zmienną o nazwie `myListener` (której typ zostaje określony jako obiekt) i umieścić w niej nowy obiekt. Do tego momentu zmienna `myListener` posiada przyporządkowany zwykły obiekt. Nie reaguje on jeszcze na zdarzenia wywoływane przez klasę `MovieClipLoader`. Jednak za chwilę go tego nauczymy.

13. Kliknij na końcu drugiego wiersza kodu i wciśnij kilka razy klawisz *Enter* (w systemie Windows) lub *Return* (w systemie Mac OS). Flash ignoruje puste wiersze, więc możesz to zrobić dowolną liczbę razy. Następnie napisz:

```
myMCL.addListener(myListener);
```

```
var myMCL:MovieClipLoader = new MovieClipLoader();
var myListener:Object = new Object();

myMCL.addListener(myListener);
```

Ten wiersz kodu przyporządkowuje obiekt nasłuchujący (listener) obiektowi klasy `MovieClipLoader` o nazwie `myMCL`. Od tego momentu za każdym razem, gdy obiekt `myMCL` wywoła jakieś zdarzenie (zacznie lub skończy ładować plik itp.), przyporządkowany mu obiekt nasłuchujący (listener) spowoduje uruchomienie określonego skryptu.

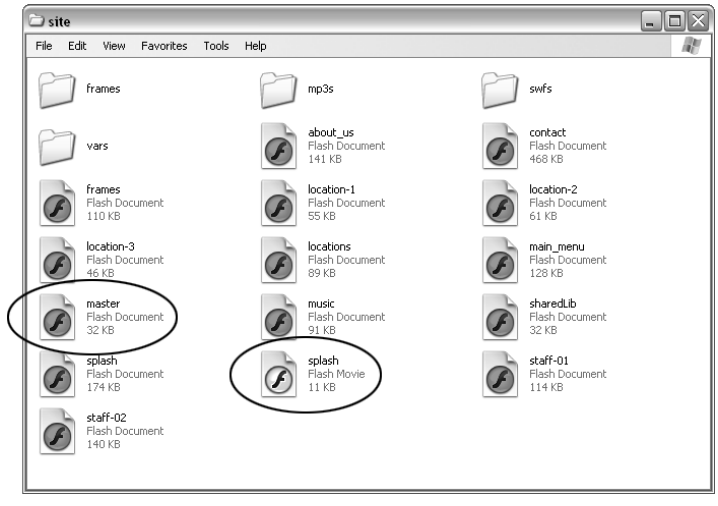
Uwaga

Możesz dodać dowolną liczbę obiektów nasłuchujących (listenerów) reagujących na zdarzenia wywoływane przez ten obiekt. Kilka z nich zobaczysz w tym i w następnych rozdziałach.

Gratuluję! Właśnie utworzyłeś pierwszy skrypt dla obiektu klasy `MovieClipLoader`! Później zobaczysz, że można z nim zrobić dużo więcej. Abyś nie zużył zbyt wielu szarych komórek naraz, kolejne możliwości będziemy jej dodawać stopniowo. W rozdziale 8. „Tworzenie preloadera" umieścimy w nim preloader. Jednak w tej chwili każemy jej tylko ładować pliki SWF i JPG. Zajmiemy się tym w następnym punkcie.

14. Ukryj lub zminimalizuj okno programu Flash MX 2004. Następnie otwórz folder *la_eyeworks*, który umieściłeś na pulpicie. Wejdź do katalogu *site* i dwukrotnie kliknij plik *splash.swf* (system operacyjny może nie pokazywać rozszerzenia pliku). Powinno to spowodować automatyczne otworzenie animacji, którą zawiera, w programie Macromedia Flash Player 7, dostarczanym wraz z programem instalacyjnym Flasha MX 2004. W następnym punkcie wpiszemy wiersz kodu, który każe obiektowi `MovieClipLoader` umieszczonemu w pliku *master.fla* załadować plik *splash.swf*.

15. Zamknij plik *splash.swf* i spójrz na zawartość folderu *site*. Zauważ, że znajduje się w nim również plik *master.fla* (w którym umieszczony jest skrypt używający klasy `MovieClipLoader`).



Po załadowaniu zasobów do projektu w programie Flash MX 2004 należy zwrócić uwagę na to, gdzie są one umieszczone względem pliku, który służy do ich załadowania.

16. W programie Flash MX 2004 upewnij się, że nadal zaznaczona jest pierwsza klatka kluczowa warstwy *a* oraz że otwarty jest panel *Actions* (*F9*). Następnie kliknij koniec ostatniego wiersza kodu i wciśnij kilka razy klawisz *Enter* (w systemie Windows) lub *Return* (w systemie Mac OS), tworząc przerwę (dzięki temu uzyskujemy przestrzeń pomiędzy ważnymi częściami skryptu).

17. Napisz:

```
myMCL.loadClip("splash.swf", 5);
```

```
var myMCL:MovieClipLoader = new MovieClipLoader();
var myListener:Object = new Object();

myMCL.addListener(myListener);

myMCL.loadClip("splash.swf", 5);
```

Ten wiersz każe obiektowi typu `MovieClipLoader` (`myMCL`) załadować plik *splash.swf* do poziomu 5. Prawdopodobnie wiesz już, że ładując go do innego pliku SWF, masz dwie możliwości: możesz umieścić go w klipie filmowym lub na określonym poziomie. Poziom jest czymś, co przypomina niewidoczną warstwę, znajdującą się nad filmem, do którego ładowana jest nowa zawartość. Umożliwia on skonfigurowanie kolejności ułożenia nowego obiektu względem istniejącej zawartości filmu. W tym przypadku plik *splash.swf* zostaje załadowany na poziom 5. Poziom 2. znajduje się nad 1., 3. nad 2. itd. Film główny, do którego plik jest ładowany, jest określany jako poziom 0 (w języku ActionScript — `_level0`).

Uwaga

W momencie ładowania zasobów (plików SWF, JPG itp.) do określonego poziomu, możesz wykorzystać dowolny poziom, od 0 do 2 130 706 429 (tak, to ponad dwa miliardy). Jak widzisz, mamy do dyspozycji olbrzymią ilość miejsca.

Uwaga

Kompatybilność wsteczna klasy MovieClipLoader

Należy pamiętać o tym, że obiekt typu `MovieClipLoader`*, taki jak ten, który właśnie utworzyłeś, nie zawsze jest kompatybilny z wcześniejszymi wersjami pluginu umożliwiającego odtwarzanie plików programu Flash. Będzie on działał tylko w najnowszych wersjach programu Flash Player. W tej książce nauczysz się tworzyć wykrywacz pluginu, który pozwoli na identyfikację użytkowników korzystających z jego wcześniejszych wersji. Przekierowując ich do innej strony lub witryny, zapobiega się otwieraniu filmu przez osoby, które nie posiadają kompatybilnej wersji pluginu.*

Jeśli chcesz (lub wymaga tego Twój klient) zapewnić kompatybilność z wcześniejszymi wersjami pluginu, musisz używać funkcji `loadMovie` *(lub* `loadMovieNum`*). Książka dotyczy najnowszej wersji programu Flash i dostępnych w niej funkcji, dlatego zawiera informacje na temat wielu funkcji, które nie są kompatybilne z wcześniejszymi wersjami pluginu. Jednak ze względu na to, że klasa* `MovieClipLoader` *jest integralną częścią metody tworzenia witryn, z której właśnie korzystamy, uznałem, że powinieneś zdawać sobie sprawę z występującego w jej przypadku braku kompatybilności wstecznej.*

Uwaga

Nowe reguły bezpieczeństwa

W programie Macromedia Flash 7 Player zostały wprowadzone nowe reguły bezpieczeństwa, które uniemożliwiają ładowanie do pliku SWF wszelkiego rodzaju danych (zmiennych, plików SWF i JPG itd.) ze źródeł znajdujących się poza domeną, w której plik ten się znajduje. Innymi słowy, jeśli Twój plik znajduje się pod adresem http://www.mojadomena.com, nie możesz do niego załadować danych z innego serwera, dostępnego pod adresem http://www.innadomena.com. Żeby to umożliwić, musisz utworzyć tzw. plik reguł międzydomenowych. Jest to prosty plik XML (o nazwie crossdomain.xml), przechowywany w głównym katalogu serwera zawierającego dane, które chcesz załadować. Znajduje się w nim lista adresów URL, które mogą korzystać z danych umieszczonych na serwerze. Oto przykładowa zawartość takiego pliku:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.macromedia.com/xml/dtds/
   cross-domain-policy.dtd">
<cross-domain-policy>
   <allow-access-from domain="www.company.com" />
</cross-domain-policy>
```

Jeśli spróbujesz załadować do projektu stworzonego w programie Flash MX 2004 dane, które znajdują się w innej domenie, nie używając pliku reguł międzydomenowych, w momencie, gdy użytkownik odwiedzi Twoją witrynę, zobaczy okno dialogowe z pytaniem, czy chce załadować dane z innego serwera. Oczywiście jako projektant nie chcesz, żeby było ono wyświetlane.

Więcej informacji na temat reguł bezpieczeństwa w pluginie Macromedia Flash 7 oraz pliku reguł międzydomenowych znajdziesz na stronie WWW firmy Macromedia (niestety, tylko w języku angielskim) pod adresem:

http://www.macromedia.com/support/flash/ts/documents/loadvars_security.htm

W najnowszej (powstałej w czasie pisania tej książki) wersji programu Macromedia Flash Player, czyli w 7.0.r19, została dodana nowa akcja (System.security.loadPolicyFile) pozwalająca na umieszczenie plików reguł międzydomenowych w innych miejscach na serwerze. Więcej na jej temat przeczytasz pod adresem:

http://www.macromedia.com/devnet/mx/flash/articles/fplayer_security.html

18. Przetestuj film, wybierając z menu polecenie *Control/Test Movie* (*Kontrola/Testuj Film*).

Jak widzisz, plik *splash.swf* zostaje bez problemu załadowany do pliku *master.swf* (w którym znajduje się skrypt wykorzystujący klasę `MovieClipLoader`). Po umieszczeniu pliku SWF lub JPG na danym poziomie, jego lewy górny róg (x:0, y:0) jest wyrównywany z lewym górnym rogiem głównego pliku SWF (również x:0, y:0). Dzięki temu plik *splash.swf* jest idealnie wyrównany z plikiem *master.swf*. Jest tak dlatego, że oba dokumenty zostały utworzone z użyciem tego samego rozmiaru sceny. W ten sposób podczas tworzenia zawartości plików (logo, muzyki, sekcji *about us* itd.) wiesz, jak wszystkie elementy zostaną ustawione względem głównego pliku *master.swf*.

Wiem, że cztery wiersze kodu wpisane w tym ćwiczeniu mogą wydawać się przesadą, biorąc pod uwagę to, że służą one tylko do załadowania pliku SWF. Pamiętaj jednak, że ten kod będzie kontrolował ładowanie wszystkich plików SWF i JPG w całej witrynie firmy L.A. Eyeworks, nie wspominając już o przewagach klasy `MovieClipLoader` nad starą funkcją `loadMovie`, którą już powinieneś znać (więcej informacji na temat różnic pomiędzy nimi znajdziesz w poprzedniej części, „Czym jest MovieClipLoader i czym różni się od loadMovie?”).

W ten sposób uzyskałeś pierwszy działający przykład wykorzystania klasy `MovieClipLoader`!

19. Zapisz plik *master.fla*, wybierając z menu polecenie *File/Save*.

W następnym ćwiczeniu nauczysz się używać niezwykle przydatnej funkcji, pozwalającej na zmniejszenie rozmiaru pliku: biblioteki współdzielonej. Jest ona plikiem SWF zawierającym garść zasobów — symboli, czcionek itp., możliwych do wykorzystania w innych plikach SWF. Jej koncepcja jest podobna do koncepcji symbolu (podobnie jak on, pozwala na wielokrotne wykorzystywanie tego samego elementu), z tą różnicą, że w jej przypadku mamy do czynienia z całą biblioteką elementów dostępnych dla wielu plików SWF.

Czym jest biblioteka Współdzielona?

Widziałeś już, jak można wykorzystywać symbole (symbole graficzne, klipy filmowe itp.) do redukcji rozmiaru pliku, w którym jest zapisany film stworzony w programie Flash MX 2004, poprzez wielokrotne wykorzystywanie identycznych elementów. A jeśli podzielimy witrynę na kilka części i będziemy je ładować używając funkcji `loadMovie` lub klasy `MovieClipLoader` (tak jak w tej książce), przy czym będziemy chcieli wykorzystać w każdym z nich ten sam symbol? Lub gdy znajdziemy się w często spotykanej sytuacji, w której wszystkie pliki będą korzystały z tej samej czcionki? Normalnie powinniśmy skopiować symbol do każdego pliku lub w przypadku czcionki osadzić ją w każdym pliku, w którym chcemy z niej korzystać. Spowodowałoby to zwiększenie rozmiaru pliku, ponieważ ten sam symbol byłby ściągany wielokrotnie, podobnie jak czcionka. Co możemy zrobić w takiej sytuacji? Nie możemy przecież udostępnić go kilku plikom SWF, prawda? Nieprawda. Możemy to zrobić używając biblioteki współdzielonej.

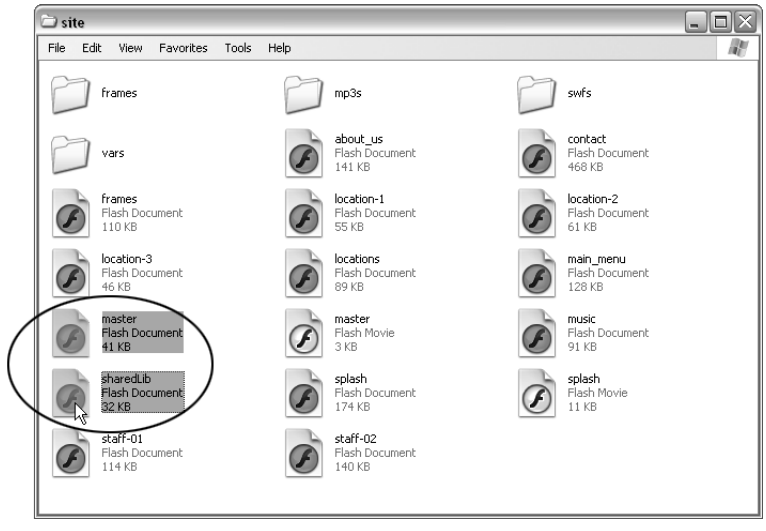
Sama biblioteka współdzielona nie jest niczym niezwykłym. Jest to zwykły plik FLA, którego biblioteka jest wypełniona symbolami, czcionkami itp. Różnica pomiędzy nim i zwykłym plikiem FLA zaczyna być widoczna w momencie przydzielania nazw poszczególnym elementom. Zrobimy to w następnym ćwiczeniu. Jeśli chcesz udostępnić jeden z elementów innemu plikowi FLA, wystarczy, że przeciągniesz go do jego biblioteki. Flash MX 2004 automatycznie utworzy łącze do elementu współdzielonego (które jest po prostu odnośnikiem do obiektu znajdującego się w bibliotece współdzielonej), pozwalając Ci na wykorzystanie go w swoim projekcie. Może on być udostępniony dowolnej liczbie plików.

Element pochodzący z biblioteki współdzielonej nie zostaje osadzony w pliku, w którym został umieszczony. Pozostaje w niej nadal. W odpowiednim pliku znajduje się jedynie odnośnik wskazujący jego położenie w bibliotece! Pod tym względem element biblioteki współdzielonej przypomina symbol (element, który można wykorzystywać wielokrotnie). Różni się od niego jedynie możliwością wielokrotnego wykorzystywania elementów w różnych plikach SWF. Czyż to nie potężne narzędzie?

Ćwiczenie 2. Tworzenie i przygotowywanie biblioteki współdzielonej

W tym ćwiczeniu utworzymy bibliotekę współdzieloną. Będzie ona zawierała elementy, które wykorzystamy w następnych ćwiczeniach.

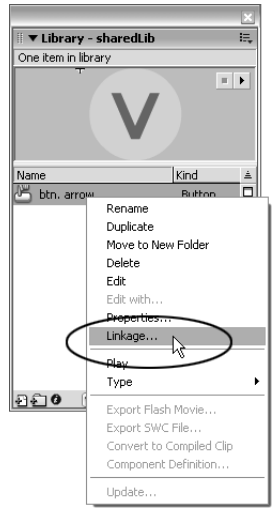
1. Otwórz plik *master.fla* (który utworzyłeś w ostatnim ćwiczeniu), jeśli go przypadkowo zamknąłeś, a następnie plik *shareLib.fla*. Oba znajdują się w folderze *site* umieszczonym w katalogu *la_eyeworks* na Twoim pulpicie (patrz pierwszy rysunek na następnej stronie).
2. Otwórz bibliotekę pliku *shareLib.fla*. Zobaczysz w niej tylko jeden obiekt — przycisk o nazwie *btn.arrow*. Nie martw się. Za chwilę umieścimy w niej więcej elementów (patrz drugi rysunek na następnej stronie).



Głównym zastosowaniem biblioteki współdzielonej jest udostępnianie zasobów możliwych do wykorzystania w wielu różnych plikach SWF. Jeśli dany element jest wykorzystywany tylko raz, w jednym pliku, umieszczanie go w takiej bibliotece nie ma sensu.

3. Kliknij prawym przyciskiem myszy (w systemie Windows) lub kliknij z wciśniętym klawiszem *Ctrl* (w systemie Mac OS) symbol *btn.arrow* znajdujący się w bibliotece *sharedLib*, a następnie wybierz z menu polecenie *Linkage* (patrz pierwszy rysunek na następnej stronie).

4. Spowoduje to otworzenie okna dialogowego *Linkage Properties*. Możesz w nim określić sposób udostępnienia elementu znajdującego się w bibliotece oraz jego nazwę. Najpierw kliknij opcję Export for ActionScript. Spowoduje to automatyczne zaznaczenie pola *Export in first frame*. Następnie wpisz w polu *Identifier* nazwę `arrowBtn`. Może ona być absolutnie dowolna. Jednak podobnie jak w przypadku nazwy instancji symbolu nie powinna się zaczynać od liczby. Nie może też zawierać spacji ani znaku specjalnego, takiego jak `?`, `*` itp. Na koniec zaznacz pole *Export for runtime sharing*. Spowoduje to automatyczne wypełnienie pola URL nazwą, którą otrzyma plik w momencie publikacji, czyli *shareLib.swf* (patrz drugi rysunek na następnej stronie).



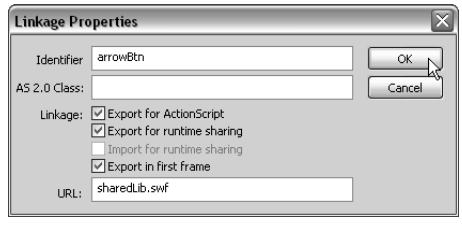


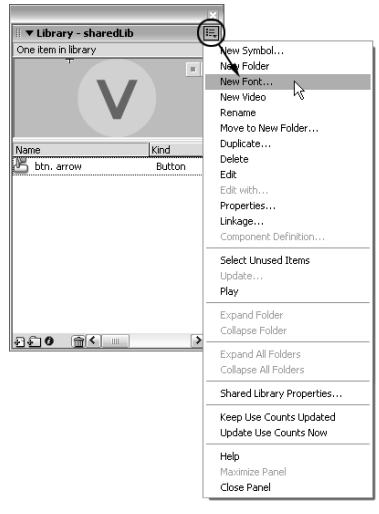
Tabela znajdująca się poniżej zawiera opis powyższych opcji.

Opcje dostepne w oknie Linkage properties

Opcja	Opis
Identifier	Jest to nazwa, którą należy przypisać każdemu z udostępnianych elementów. Dopuszczalny jest dowolny ciąg znaków. Nie powinien on jednak zawierać spacji ani znaków specjalnych, takich jak ?, # itp. Nie może również zaczynać się cyfrą.
AS 2.0 Class	Nazwa klasy w języku ActionScript 2.0, która ma zostać przyporządkowana do symbolu.
Export for ActionScript	Pozwala na manipulowanie elementem za pomocą języka ActionScript.
Export for runtime sharing	Pozwala na wykorzystanie elementu w czasie odtwarzania pliku SWF. Jeśli nie zaznaczysz tej opcji, element nie pojawi się w opublikowanym dokumencie.
Import for runtime sharing	Ta opcja jest dostępna po wybraniu w dokumencie docelowym polecenia *Linkage* dla obiektu pochodzącego z biblioteki współdzielonej. Jej wyłączenie nie pozwala na wyświetlanie elementu w czasie odtwarzania.
Export in first frame	Powoduje eksport elementu biblioteki współdzielonej w sposób, który umożliwia korzystanie z niego już w pierwszej klatce filmu. Jeśli nie zaznaczysz tej opcji, będziesz mógł umieścić jego instancję na scenie, w klatce, w której będziesz go potrzebował.
URL	To pole zawiera adres w formacie URL, pod którym znajduje się plik SWF. Flash MX 2004 automatycznie wpisuje do niego nazwę pliku, w którym znajduje się biblioteka współdzielona. W tym przypadku jest to *sharedLib.swf*.

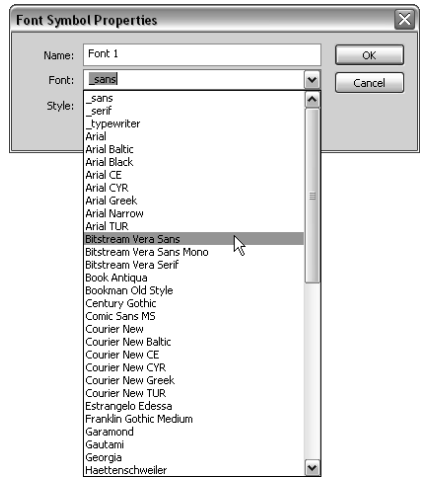
5. Kliknij *OK*. Symbol jest teraz współdzielony i możesz go umieszczać w bibliotekach innych plików FLA.

   Teraz powinniśmy utworzyć symbol czcionki, która ma być udostępniona innym plikom SWF.



6. Kliknij przycisk *Library Preferences* (właściwości biblioteki), a następnie wybierz z menu polecenie *New Font* (nowa czcionka).

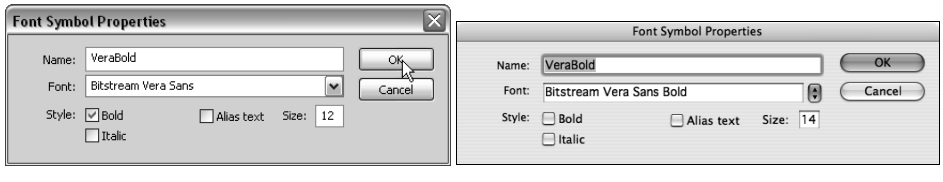
   Spowoduje to otworzenie okna dialogowego *Font Symbol Properties* (właściwości symbolu czcionki). Możesz w nim wybrać czcionkę, której symbol chcesz utworzyć, i nadać jej nazwę, która będzie widoczna w bibliotece. Na liście powinieneś widzieć czcionki *Bitstream*, które zainstalowałeś w rozdziale 2. „Od czego powinienem zacząć?". W systemie Mac OS X zobaczysz więcej czcionek *Bitstream Vera* niż na poniższej ilustracji pochodzącej z systemu Windows. Windows grupuje różne style, na komputerach Macintosh są one wyświetlane oddzielnie. Oznacza to, że użytkownicy tych dwóch systemów będą musieli użyć różnych procedur. Wspomnę o tym w następnych punktach.



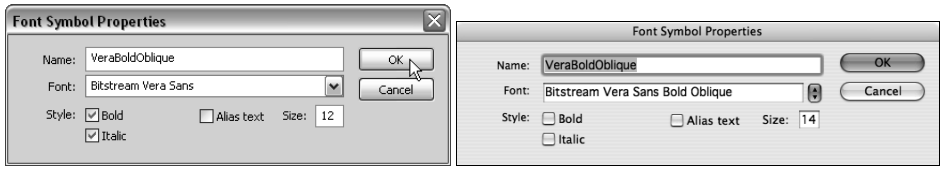
7. Niezależnie od tego, z którego systemu korzystasz, wybierz z rozwijanego menu czcionkę *Bitstream Vera Sans*. W polu *Name* wpisz `Vera`. Upewnij się, że nie są zaznaczone opcje *Bold*, *Italic* i *Alias text*. Rozmiar tekstu nie jest istotny w przypadku sposobu, w jaki wykorzystujemy czcionki w tej książce. Dlatego pozostaw go bez zmian i kliknij *OK*.

Wspaniale! Właśnie utworzyłeś symbol w bibliotece. Jest nim cała czcionka! Nieźle. Następnie dodamy trzy inne czcionki *Vera*, których będziemy używać do tworzenia witryny firmy L.A. Eyeworks.

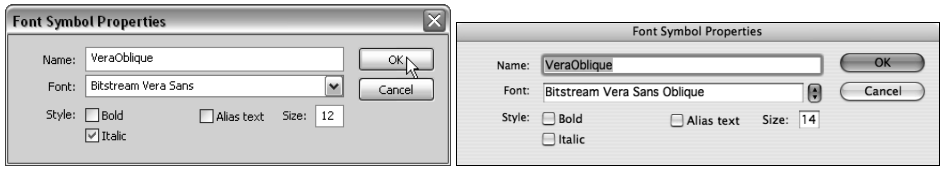
8. Powtórz punkt 7., ale w systemie Windows wybierz czcionkę *Bitstream Vera Sans*, a następnie kliknij opcję *Bold*. W systemie Mac OS wybierz *Bitstream Vera Sans Bold*. Nie musisz zaznaczać opcji *Bold*. Niezależnie od tego, z którego systemu korzystasz, wpisz w polu *Name* nazwę `VeraBold` i kliknij *OK*.



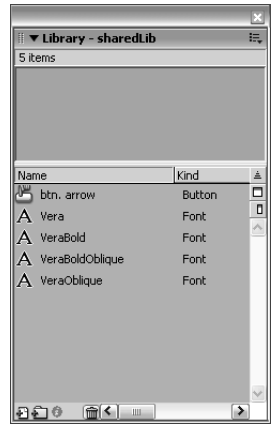
9. Jeszcze raz powtórz punkt 7., ale w systemie Windows wybierz czcionkę *Bitstream Vera Sans*, a następnie kliknij opcje *Bold* i *Italic*. W systemie Mac OS wybierz *Bitstream Vera Sans Bold Oblique*. Nie musisz zaznaczać opcji *Bold* i *Italic*. Niezależnie od tego, z którego systemu korzystasz, wpisz w polu *Name* nazwę `VeraBoldOblique` i kliknij *OK*.



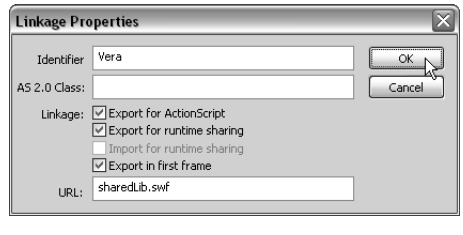
10. Jeszcze raz powtórz punkt 7., ale w systemie Windows wybierz czcionkę *Bitstream Vera Sans*, a następnie kliknij opcję *Italic*. W systemie Mac OS wybierz *Bitstream Vera Sans Bold Oblique*. Nie musisz zaznaczać opcji *Italic*. Niezależnie od tego, z którego systemu korzystasz, wpisz w polu *Name* nazwę `VeraOblique` i kliknij *OK*.



Super! Teraz gdy zajrzysz do biblioteki pliku *sharedLib.fla*, zobaczysz, że poza przyciskiem o nazwie *btn.arrow* znajdują się w niej również cztery czcionki: *Vera*, *VeraBold*, *VeraBoldOblique* i *VeraOblique*. Musisz teraz ustawić dla każdej z nich odpowiednie opcje w oknie *Linkage Properties* (tak jak dla symbolu *btn.arrow* w punktach 3. i 4.), dzięki czemu będą one dostępne dla innych plików SWF.



11. W bibliotece *sharedLib* kliknij prawym przyciskiem myszy (w systemie Windows) lub kliknij z wciśniętym klawiszem *Ctrl* (w systemie Mac OS) symbol czcionki *Vera* i wybierz z rozwijanego menu polecenie *Linkage*. W oknie Linkage Properties zaznacz opcje: *Export for ActionScript* (która spowoduje automatyczne zaznaczenie pola Export in first frame) i *Export for runtime sharing*. Upewnij się również, że w polu *Identifier* automatycznie pojawił się napis *Vera* (czyli nazwa symbolu czcionki), a w polu URL nazwa pliku *sharedLib.swf*. Kliknij *OK*.



12. Powtórz powyższy proces dla pozostałych trzech czcionek. Wszystkie ustawienia powinny być takie same jak w kroku 11., z następującymi wyjątkami: dla czcionki *VeraBold* wpisz w polu *Identifier* identyfikator `VeraBold`, dla czcionki *VeraBoldOblique* — `VeraBoldOblique`, a dla *VeraOblique* — `VeraOblique`. Uf!

Teraz, gdy skonfigurowałeś połączenia z symbolami, możesz ich użyć w dowolnej liczbie plików SWF (w następnym rozdziale zobaczysz, jak to zrobić). Dużą korzyścią płynącą z korzystania z biblioteki współdzielonej jest to, że same czcionki są osadzone tylko w pliku *sharedLib*. Wszystkie pozostałe elementy witryny mogą z nich korzystać, ale nie muszą ich przechowywać. Po prostu wykorzystują one referencje do czcionek w pliku *sharedLib.swf*. Pozwala to na znaczną redukcję rozmiaru projektu, ponieważ nie ma konieczności osadzania czcionek w każdym pliku, w którym są one użyte. Hura!

Ostatnią rzeczą, którą musimy zrobić, jest mały trik związany z biblioteką współdzieloną. Jednak zanim zaczniemy, powinieneś zrozumieć, dlaczego istnieje taka konieczność. W następnym rozdziale użyjemy niektórych czcionek w innych plikach SWF. Gdy użytkownik odwiedzi naszą witrynę i zacznie przeglądać jej zawartość, to w momencie, gdy po raz pierwszy będzie wymagany element pochodzący z biblioteki, Flash zacznie ściągać cały plik, który go zawiera. Niestety nie ściąga tylko jednego elementu, lecz całą bibliotekę. I choć nie stanowi to dużego problemu, to jednak w przypadku sporej biblioteki może znacznie spowolnić odtwarzanie filmu lub spowodować jego zatrzymanie. Najlepiej byłoby oczywiście załadować plik *sharedLib.swf* w sposób kontrolowany, aby użytkownik wiedział, co się dzieje.

Problem w tym, że zwykle wcześniejsze załadowanie pliku *sharedLib.swf* nic nie da (choć logika sugeruje, że powinno). Program Flash MX 2004 po prostu załaduje go ponownie w momencie, gdy użyty zostanie pierwszy element współdzielony. Zamiast tego utworzymy plik SWF, którego zadaniem będzie rozpoczęcie ściągania biblioteki w momencie, w którym będziemy mogli je kontrolować. Będzie on zawierał tylko jeden element biblioteki współdzielonej. Plik *trigger.swf* będzie ładowany na początku odtwarzania projektu, z użyciem klasy `MovieClipLoader`. Ponieważ znajduje się w nim element współdzielony, spowoduje to rozpoczęcie ściągania pliku *sharedLib.swf*. Dzięki temu od tego momentu cała biblioteka współdzielona będzie dostępna dla wszystkich plików SWF zawartych w witrynie.

Po pierwsze, musimy utworzyć plik uruchamiający, o nazwie *trigger.fla*, który wykorzysta jeden z elementów biblioteki do rozpoczęcia ściągania pliku *sharedLib.swf*.

13. Zapisz zmiany w pliku *sharedLib.fla*, a następnie wybierz z menu polecenie *File/Publish* (plik/publikuj), aby wyeksportować plik SWF do katalogu, w którym znajduje się biblioteka (*Pulpit\la_eyeworks\site*).

14. Zamknij plik *sharedLib.fla*.

15. Utwórz nowy, pusty plik FLA, wciskając *Ctrl+Alt+N* (w systemie Windows) lub *Cmd+Opt+N* (w systemie Mac OS).

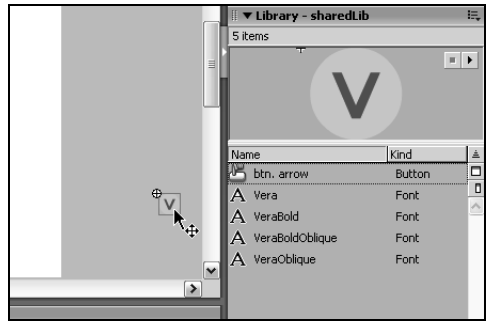
Nie musisz się przejmować rozmiarami ani kolorem tła tego pliku, ponieważ użyjemy go tylko do uruchomienia ładowania biblioteki współdzielonej. Użytkownik nawet nie będzie wiedział o tym, że zostaje on załadowany. Po zakończeniu ładowania biblioteki plik *trigger.swf* zostanie zastąpiony plikiem *splash.swf*.

16. Najpierw zapisz nowy plik FLA, wybierając z menu polecenie *File/Save As*. Umieść go w folderze *Pulpit\la_eyeworks\site*, nadając mu nazwę `trigger`. Kliknij przycisk *Save*.

17. Otwórz bibliotekę *sharedLib*, wybierając z menu polecenie *File/Import/Open External Library* (plik/importuj/otwórz bibliotekę zewnętrzną). W oknie dialogowym *Open as Library* dwukrotnie kliknij plik *sharedLib*. Spowoduje to otworzenie biblioteki pliku *sharedLib.fla*.

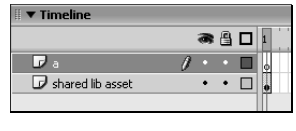
Plik uruchamiający wymaga tylko jednego symbolu z biblioteki współdzielonej. Po załadowaniu pliku *trigger.swf* i stwierdzeniu, że na scenie znajduje się taki element, rozpocznie ściąganie (i buforowanie) całej biblioteki!

18. Przeciągnij przycisk *btn.arrow* do przestrzeni roboczej (szarego obszaru wokół sceny) pliku *trigger.fla*. Powinien się znajdować poza sceną, ponieważ ma być niewidoczny po załadowaniu dokumentu do pliku *master.swf*.



Problem z plikiem *master.fla* polega na tym, że obecne ustawienie jego listwy czasowej powoduje ładowanie wszystkich plików w pierwszej klatce kluczowej. Teraz, gdy utworzyłeś plik, który powoduje załadowanie biblioteki współdzielonej, chcesz, żeby zakończyło się ono przed wyświetleniem zawartości pliku *splash.swf*. Ponieważ biblioteka jest bardzo ważnym elementem witryny (w końcu zawiera czcionki, które będą używane niemal we wszystkich plikach SWF), powinna być ładowana jako pierwsza. Dlatego powinniśmy wstawić przerwy pomiędzy ściąganiem poszczególnych plików. Zawartość początkowa (*trigger.swf*, *sharedLib.swf* itp.) zostanie załadowana od razu w pierwszej klatce kluczowej. Następnie głowica przejdzie do klatki 10., w której nastąpi ładowanie następnego elementu, logo o nazwie *splash.swf*. Musisz jednak wstawić akcję, która rozpocznie odtwarzanie pliku *master.swf* po zakończeniu ładowania plików *trigger.swf* i biblioteki współdzielonej, czyli pliku *sharedLib.swf*.

19. Zmień nazwę pierwszej warstwy na *shared lib asset*. Następnie utwórz nową warstwę i nazwij ją `a`.



20. Zaznacz pierwszą klatkę kluczową warstwy a i otwórz panel *Actions* (*F9*).

21. Wstaw akcję, która rozpocznie odtwarzanie listwy czasowej pliku *master.swf*, poprzez kliknięcie panelu *Actions* i umieszczenie w nim instrukcji:

```
_level0.play();
```

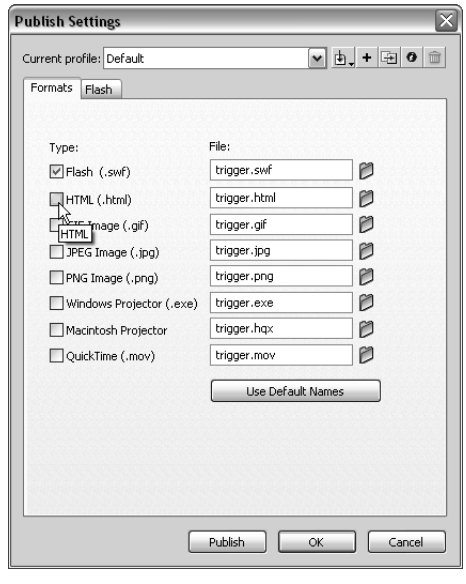
```
_level0.play();
```

Powoduje ona rozpoczęcie odtwarzania listwy czasowej pliku *master.swf* (`_level0`). Oczywiście jest ona wykonywana po całkowitym ściągnięciu pliku *trigger.swf*. Ale ponieważ plik ten zawiera również element pochodzący z biblioteki współdzielonej, przez co rozpoczyna ściąganie pliku *sharedLib.swf*, akcja zostanie wykonana dopiero w chwili, gdy oba pliki zostaną załadowane. Uf.

Teraz musimy tylko zapisać zmiany w pliku i wyeksportować go do formatu SWF.

Aby uniknąć zaśmiecania folderu *site*, powinieneś wyłączyć tworzenie dodatkowego pliku HTML w momencie publikacji pliku FLA.

22. Wybierz z menu polecenie *File/Publish Settings* (*Plik/Ustawienia publikacji*). W oknie dialogowym wyłącz opcję HTML. Dzięki temu zapobiegniesz tworzeniu pliku *trigger.html* w momencie publikacji dokumentu. Jest on niepotrzebny, ponieważ plik *trigger.swf* będzie ładowany bezpośrednio do pliku *master.swf*.



23. Kliknij przycisk *Publish*. Spowoduje to opublikowanie pliku *trigger.swf* w folderze *site* (w miejscu, w którym znajduje się plik *trigger.fla*). Następnie kliknij *OK*, aby zamknąć okno dialogowe *Publish Settings*.

24. Zapisz dokonane zmiany, wybierając z menu polecenie *File/Save*. Następnie zamknij plik *trigger.fla*, używając polecenia *File/Close* (możesz również użyć skrótu klawiszowego *Ctrl+W* w systemie Windows lub *Cmd+W* w systemie Mac OS).

Tymczasem na listwie czasowej pliku *master.fla* wszystkie akcje nadal znajdują się w pierwszej klatce kluczowej. W następnych kilku krokach przeniesiemy moment, w którym ładowany jest plik *splash.swf*, do jej dalszej części. Każemy również obiektowi `MovieClipLoader`, znajdującemu się w pliku *master.fla*, załadować plik *trigger.swf*. Oczywiście posłuży on do rozpoczęcia ładowania biblioteki współdzielonej poprzez użycie jednego z elementów, które zawiera. Na koniec stworzymy również komunikat o ładowaniu filmu, aby użytkownik nie musiał patrzeć na pustą stronę w czasie, gdy ściągane są pliki *trigger.swf* i *sharedLib.swf*.

25. Upewnij się, że masz otwarty plik *master.fla* (jeśli go zamknąłeś, możesz otworzyć go ponownie, wchodząc do folderu *Pulpit\la_eyeworks\site* i dwukrotnie klikając jego nazwę). Zaznacz pierwszą klatkę kluczową warstwy a i otwórz panel *Actions*.

26. Kliknij koniec wiersza `myMCL.loadClip("splash.swf", 5);` i wciśnij *Enter* (w systemie Windows) lub *Return* (w systemie Mac OS), aby przejść do następnej linii. Następnie załaduj plik *trigger.swf* do poziomu 5., wpisując:

```
myMCL.loadClip("trigger.swf", 5);
```

```
var myMCL:MovieClipLoader = new MovieClipLoader();
var myListener:Object = new Object();

myMCL.addListener(myListener);

myMCL.loadClip("splash.swf", 5);
myMCL.loadClip("trigger.swf", 5);
```

Ponieważ plik *trigger.swf* nie ma żadnych widocznych elementów (zawiera jedynie przycisk z biblioteki współdzielonej, umieszczony poza obszarem roboczym), po jego załadowaniu scena pliku *master.swf* będzie pusta. Gdy program Macromedia Flash Player 7 wykryje, że plik *trigger.swf* używa referencji do elementu pochodzącego z biblioteki współdzielonej, rozpocznie ściąganie (i buforowanie) jej zawartości. Po kompletnym ściągnięciu pliku, który ją zawiera (*sharedLib.swf*), będziesz mógł użyć dowolnego, pochodzącego z niej elementu. Super!

Prawdopodobnie zauważyłeś, że plik *trigger.swf* jest ładowany na ten sam poziom, co plik *splash.swf*, czyli `_level5`. Dzieje się tak dlatego, że służy on jedynie do ściągnięcia biblioteki współdzielonej. Gdy skończy, nie jest już potrzebny. Wtedy zostaje usunięty, a w jego miejsce jest ładowany plik *splash.swf*. To ma być wdzięczność? ;-) Choć ładowanie obu dokumentów do tego samego poziomu nie jest problemem, to zrobienie tego w tym samym czasie może już nim być. Przy obecnym skrypcie po rozpoczęciu ładowania pliku *splash.swf* zostaje on od razu usunięty i zastąpiony plikiem *trigger.swf*. W ten sposób użytkownik nigdy nie zobaczy pierwszego z nich. Plik *splash.swf* powinien więc być ładowany po całkowitym ściągnięciu i zakończeniu działania pliku *trigger.swf* i co za tym idzie, po załadowaniu biblioteki współdzielonej. W następnych kilku punktach zajmiemy się zmianą tej kolejności.

27. Kliknij klatkę 10. warstwy `a` i wstaw nową klatkę kluczową, wciskając *F6*. Następnie, aby była w niej również widoczna warstwa `bg`, zaznacz jej klatkę 10. i dodaj dodatkowe klatki, wciskając klawisz *F5*.

W klatce kluczowej z numerem 10, znajdującej się na warstwie a, wstawimy akcję, która rozpocznie ładowanie pliku *splash.swf*. Powód, dla którego będzie on ładowany dopiero wtedy, jest taki, że chcemy, żeby był on ładowany po całkowitym ściągnięciu pliku *trigger.swf* (które ma miejsce w klatce 1.) i biblioteki współdzielonej. Gdy to nastąpi, wykonana zostanie akcja `_level0.play();`, przyporządkowana pierwszej klatce kluczowej pliku *trigger.swf*, po czym w klatce 10. zostanie odtworzony plik *master.swf*. Oczywiście musi on zostać zatrzymany w jego pierwszej klatce kluczowej, więc w następnych kilku punktach dodamy również akcję, która to spowoduje. W rozdziale 12. „Menu główne" poznasz kolejny powód, dla którego rozpoczęcie ładowania pliku *splash.swf* jest tak ważne.

28. Kliknij klatkę kluczową z numerem 1, znajdującą się na warstwie `a` i otwórz panel *Actions* (*F9*). Następnie zaznacz wiersz `myMCL.loadClip("splash.swf", 5);`, przeciągając po nim myszą i klikając jej prawym przyciskiem (w systemie Windows) lub z wciśniętym klawiszem *Ctrl* (Mac). W menu, które się pojawi, wybierz polecenie *Cut*. Spowoduje to wycięcie akcji z klatki numer 1 i umieszczenie jej w schowku, co umożliwi jej późniejsze wklejenie w innym miejscu.

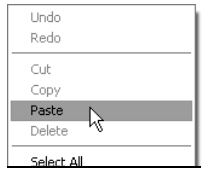
```
var myMCL:MovieClipLoader = new MovieClipLoader();
var myListener:Object = new Object();

myMCL.addListener(myListener);

myMCL.loadClip("splash.swf", 5);
myMCL.loadClip("trigger.swf", 5
```

Undo
Redo
Cut
Copy
Paste

29. Zaznacz klatkę kluczową z numerem 10 (którą dodałeś w punkcie 27.), znajdującą się na warstwie `a` i otwórz panel *Actions*. Następnie kliknij prawym przyciskiem myszy (w systemie Windows) lub z wciśniętym klawiszem *Ctrl* (w systemie Mac OS) i z wybierz z rozwijanego menu polecenie *Paste*. Spowoduje to wklejenie akcji `myMCL.loadClip("splash.swf", 5);` pochodzącej z klatki 1. do klatki 10.



Ponieważ nie chcemy, żeby te akcje były wykonywane w sposób ciągły, co może nastąpić po zapętleniu głowicy, musimy umieścić w klatce 10. również akcję `stop()`.

30. Kliknij na lewo od wiersza `myMCL.loadClip("splash.swf", 5);` i wciśnij *Enter* (w systemie Windows) lub *Return* (w systemie Mac OS), aby wstawić nad nim przerwę. Następnie kliknij pierwszy pusty wiersz w panelu *Actions* i wpisz akcję:

```
stop();
```

```
stop();
myMCL.loadClip("splash.swf", 5);
```

Gdy głowica dojdzie do klatki 10., film zostanie zatrzymany. Wtedy obiekt typu `MovieClipLoader` załaduje do poziomu 5. plik *splash.swf*, zastępując nim plik *trigger.swf* załadowany w klatce 1. Aby nie rozpoczęło się automatyczne odtwarzanie pliku *master.swf* (chcemy, żeby nastąpiło ono po kompletnym załadowaniu poprzednich plików), musimy umieścić w klatce 1. jeszcze jedną akcję `stop()`.

31. Kliknij klatkę 1. warstwy `a`. Następnie kliknij myszą przed pierwszym wierszem kodu, `var myMCL:MovieClipLoader = new MovieClipLoader();` i wciśnij *Enter* (w systemie Windows) lub *Return* (w systemie Mac OS), aby wstawić nowy wiersz. Kliknij go i dodaj akcję:

```
stop();
```

```
stop();
var myMCL:MovieClipLoader = new MovieClipLoader();
var myListener:Object = new Object();

myMCL.addListener(myListener);

myMCL.loadClip("trigger.swf", 5);
```

Teraz odtwarzanie pliku *master.swf* zostanie zatrzymane aż do momentu, gdy pliki *trigger.swf* i *sharedLib.swf* zostaną całkowicie ściągnięte i uruchomią akcję, która sprawi, że rozpoczęcie odtwarzania pliku *master.swf* stanie się możliwe:

```
_level0.play();
```

Choć jeszcze nie skończyliśmy, jest to dobry moment na przetestowanie dotychczasowych wyników pracy.

32. Zapisz zmiany dokonane w pliku *master.fla*, wybierając z menu polecenie *File/Save*. Następnie przetestuj film, klikając polecenie *Control/Test Movie*.

Gdy pojawi się okno *Preview*, w ciągu połowy sekundy powinieneś zobaczyć logo firmy. Choć dobrze jest zobaczyć, że działa ono prawidłowo, tak naprawdę nie wiesz, czy biblioteka została załadowana i czy plik *trigger.swf* właściwie wykonał swoje zadanie. Aby dowiedzieć się więcej na temat procesu ładowania oraz tego, jak może wyglądać witryna, jeśli użytkownik posiada wolny modem, użyjemy narzędzia *Bandwidth Profiler* wraz z funkcją o nazwie *Simulate Download* (w poprzednich wersjach programu Flash funkcja ta była nazwana *show streaming*).

33. W czasie, gdy nadal oglądasz podgląd pliku *master.swf*, wybierz z menu polecenie *View/Bandwidth Profiler*. Spowoduje to podzielenie okna na dwie części. Dolna zawiera podgląd pliku SWF, górna — narzędzie *Bandwidth Profiler*.

Narzędzie *Bandwidth Profiler*, jak już zapewne wiesz, pozwala na śledzenie ilości danych wysyłanych w każdej klatce w sytuacji, gdy użytkownik ogląda film przez internet. Pozwala również na sprawdzenie, jak będzie przebiegało ściąganie pliku przy różnych szybkościach łącza internetowego. W przypadku użycia akcji `loadMovie`, lub jak w naszym przypadku klasy `MovieClipLoader`, możliwe jest również sprawdzenie, które zasoby są w danym momencie ściągane i ile czasu zajmie ich ładowanie.

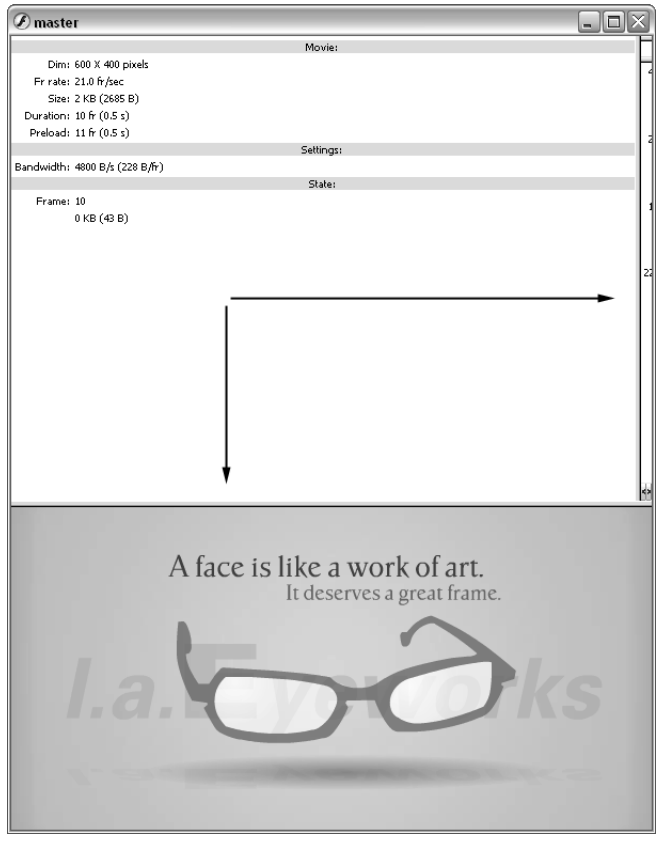
Najpierw musimy udostępnić oknu narzędzia *Bandwidth Profiler* więcej przestrzeni, abyśmy mogli cokolwiek zobaczyć.

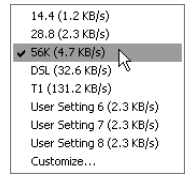
34. Powiększ okno *Preview* (podgląd) na całą wysokość monitora. Następnie przeciągnij poziomy pasek w dół tak, abyś widział zawartość pliku SWF, ale miał więcej przestrzeni dla okna narzędzia *Bandwidth Profiler*. Później przesuń pionowy pasek jak najdalej w prawo. W ten sposób zmienisz układ okien w sposób, który pozwoli na uzyskanie możliwie największej przestrzeni dla wyświetlania statystyk. Właśnie na niej będziesz mógł zobaczyć, które zasoby są ładowane i jaki jest postęp w ich ściąganiu (patrz pierwszy rysunek na następnej stronie).

Teraz, gdy ustawiłeś okno *Preview*, musisz wybrać szybkość modemu dla naszego testu.

35. Wybierz z menu polecenie *View/Download Settings/56K (4.7 KB/s)*. Jak widzisz, masz do wyboru kilka szybkości, które możesz przetestować. Możesz również utworzyć własną, wybierając polecenie *Customize*. Wybrałeś jednak najczęściej spotykaną szybkość połączenia wąskopasmowego: modem 56k. Teraz, gdy każesz Flashowi odtwarzać plik SWF, będzie on udawał, że ściąga go z internetu przez modem 56k. Świetnie! (Patrz drugi rysunek na następnej stronie).

36. Wybierz z menu polecenie *View/Simulate Download*.

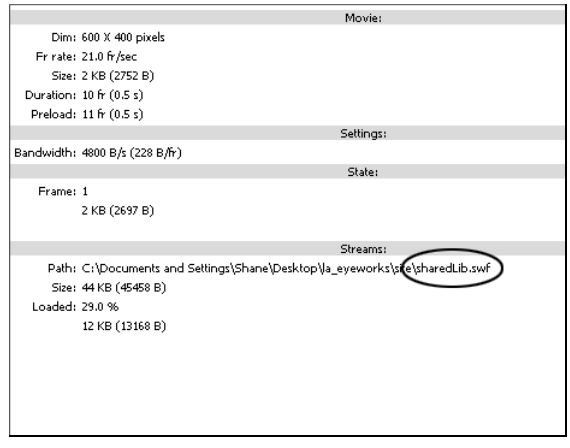


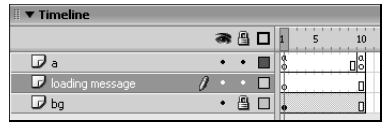


Obserwuj okno *Bandwidth Profiler*. Będą w nim pokazane ściągane zasoby oraz postęp w ich ładowaniu. Zobaczysz, że plik *sharedLib.swf* zaczyna być ściągany niemal natychmiast. Po jego załadowaniu, w sekcji *State* zaobserwujesz szybkie przejście do klatki 10. i odtworzenie pliku *splash.swf*. Niesamowite! Jedynym problemem związanym z aktualnym ustawieniem jest to, że użytkownik w czasie ładowania w tle pliku *sharedLib.swf* musi siedzieć i patrzeć na pusty ekran. W tym przypadku biblioteka współdzielona jest stosunkowo mała (44 KB), jednak nie zawsze tak jest. Aby użytkownik wiedział, co się dzieje, powinieneś umieścić w filmie komunikat, który będzie go o tym informował. Dzięki temu nikt nie będzie myślał, że witryna nie działa (patrz pierwszy rysunek na następnej stronie).

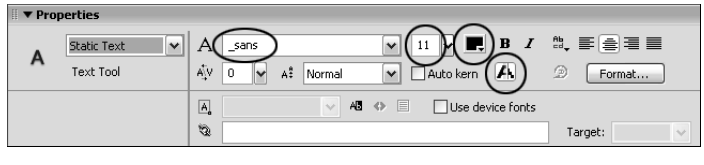
37. Zamknij okno *Preview*.

38. Kliknij najniższą warstwę, o nazwie `bg`, i dodaj nową. Zmień jej nazwę na `loading message`. Jest to warstwa, która będzie zawierała tekst `loading assets` (patrz drugi rysunek na następnej stronie).





39. Wybierz narzędzie *Text* i kliknij warstwę `loading message`. Następnie w oknie *Properties Inspector* wybierz opcję *Static Text*, czcionkę *_sans* (znajdującą się na samej górze listy czcionek w systemie Windows lub na samym dole w systemie Mac OS), rozmiar 11 i kolor czarny. Upewnij się, że zaznaczone są opcje *Align Center* i *Alias Text*. Pozwalają one na wpisanie komunikatu z użyciem czcionki systemowej, *_sans*. Jak już zapewne wiesz, nie jest ona osadzana w publikowanym pliku SWF, lecz każe komputerowi wykorzystać czcionkę, która jest już zainstalowana w systemie. Podobnie działa wyświetlanie tekstu na stronach WWW opartych na języku HTML.



Uwaga

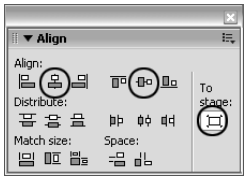
W programie Flash MX 2004 znajduje się błąd związany z narzędziem Text. Czasem (piszę „czasem", ponieważ nie zdarza się to zawsze) po jego wybraniu zawartość okna Properties Inspector nie jest aktualizowana i nie pozwala na wybranie związanych z nim opcji. Jeśli zdarzy Ci się taka sytuacja, to aby mieć do nich dostęp, wybierz narzędzie Text i kliknij myszą na scenie. Choć spowoduje to utworzenie pola tekstowego opartego na poprzednich ustawieniach, zaktualizuje również zawartość okna Properties Inspector. Wówczas będziesz mógł usunąć pole i ustawić odpowiednie opcje.

Chcemy użyć czcionki systemowej, ponieważ dzięki temu tekst komunikatu zostanie wyświetlony od razu, bez potrzeby ściągania dodatkowych czcionek. W ten sposób już od momentu załadowania pliku użytkownik będzie wiedział, co się dzieje.

40. Kliknij na scenie i napisz `loading assets`. Jest to krótki komunikat, który użytkownik będzie widział w czasie oczekiwania na zakończenie ładowania pliku *sharedLib.swf*.

41. Wciśnij klawisz *Esc*, aby wyjść z trybu edycji tekstu i otwórz panel *Align* (wyrównaj, *Ctrl+K* w systemie Windows lub *Cmd+K* w systemie Mac OS).

42. Upewnij się, że włączony jest przycisk *To Stage*, a następnie kliknij opcję *Align horizontal center* (wyrównaj do środka w poziomie) i *Align vertical center* (wyrównaj do środka w pionie). Spowoduje to ustawienie tekstu dokładnie na środku sceny.



43. Zamknij panel *Align*.

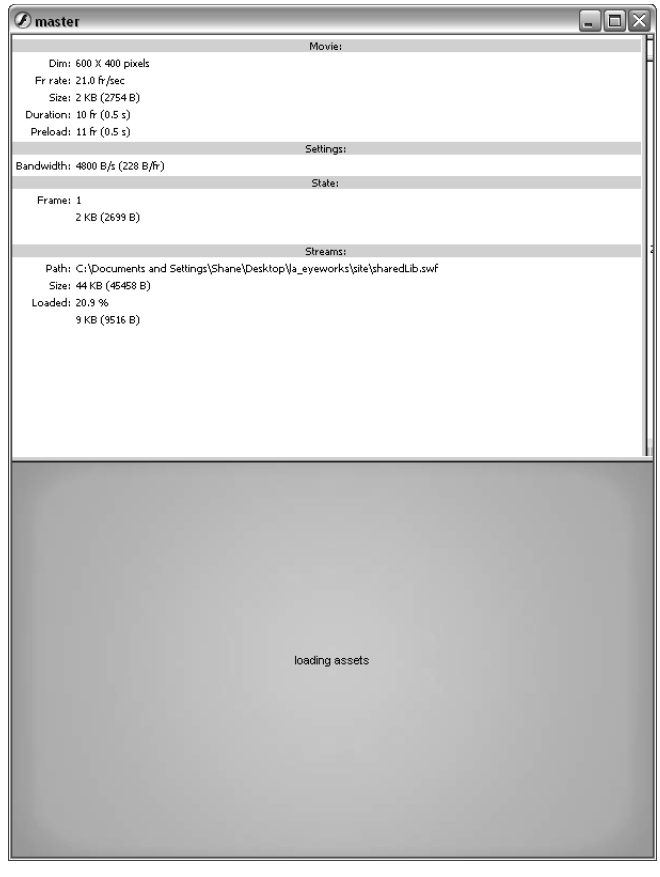
Chcemy, żeby komunikat `loading assets` był wyświetlany tylko w momencie, gdy głowica zatrzyma się na pierwszej klatce kluczowej. Po odtworzeniu klatki 10. i załadowaniu pliku *splash.swf* powinien on zniknąć. Aby tak się stało, musisz wstawić pustą klatkę kluczową w miejscu klatki 2. warstwy `loading message`.

44. Kliknij klatkę 2. warstwy `loading message` i umieść w niej pustą klatkę kluczową, wciskając *F7*. Teraz komunikat `loading assets` jest widoczny tylko w klatce 1.

45. Zablokuj warstwy `loading message` i `a`, a następnie zapisz zmiany w pliku *master.fla*, wybierając z menu polecenie *File/Save*.

46. Ponownie przetestuj film, wybierając z menu polecenie *Control/Test Movie*. Gdy pojawi się okno *Preview*, wybierz z menu polecenie *View/Simulate Download*, aby zobaczyć, jak będzie przebiegało ściąganie pliku *master.swf* i ładowanych do niego zasobów. Po jego całkowitym załadowaniu komunikat zniknie, a zamiast niego pojawi się logo firmy, czyli plik *splash.swf* (patrz rysunek na następnej stronie).

Gratulacje! Właśnie utworzyłeś, załadowałeś i udostępniłeś komunikat informujący o ładowaniu biblioteki współdzielonej. Teraz, gdy zostanie ona załadowana do bufora, będziesz mógł jej natychmiast użyć w dowolnym pliku SWF, ładowanym do pliku *master.swf*.



W następnym rozdziale nauczysz się nie tylko korzystać z elementów pochodzących z biblioteki współdzielonej, ale też zapełniać dynamiczne pole tekstowe tekstem pochodzącym z zewnętrznego pliku TXT (za pomocą klasy `LoadVars`). Dowiesz się również, jak umożliwić przewijanie tekstu oraz jak go modyfikować używając kombinacji języka HTML i kaskadowych arkuszy stylów (CSS), a także jak wstawiać do niego pliki SWF (kolejna nowa funkcja)! W następnym rozdziale znajduje się cała masa nowych, ekscytujących przykładów, więc zamknij drzwi i powiedz swojej drugiej połowie, że dziś musi iść spać bez Ciebie, ponieważ masz na ten wieczór inne plany! ;-)