

» Idź do

- Spis treści
- Przykładowy rozdział

» 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

» Kontakt

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl
© Helion 1991-2008

ActionScript 3.0 dla Adobe Flash CS4/CS4 PL Professional. Oficjalny podręcznik

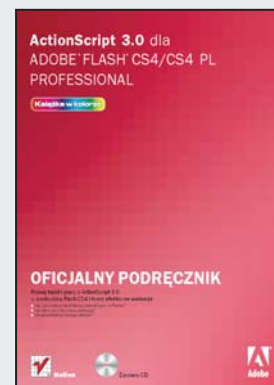
Autor: Adobe Creative Team

Tłumaczenie: Paweł Koronkiewicz

ISBN: 83-7197-641-0

Tytuł oryginału: [ActionScript 3.0 for Adobe Flash CS4 Professional Classroom in a Book](#)

Format: 170×230, stron: 368



Poznaj tajniki pracy z ActionScript 3.0 w środowisku Flash CS4 i twórz efektowne animacje

- Jak wprowadzać mechanizmy interakcyjne we Flashu?
- Jak sterować osią czasu i animacją?
- Jak pisać funkcje obsługi zdarzeń?

Adobe ActionScript 3.0 to zaawansowany język programowania, w pełni zintegrowany z Flashem CS4 – rozbudowanym środowiskiem projektowania graficznego, wyposażonym w narzędzia do pracy z dwu- i trójwymiarową animacją, dźwiękiem, grafiką wektorową i bitmapową, tekstem oraz obrazem wideo. Dzięki temu Adobe ActionScript 3.0 umożliwia tworzenie multimedialnych, bogatych i w pełni interaktywnych aplikacji, takich jak gry, systemy do e-nauki i e-commerce oraz tradycyjne aplikacje dla platformy Adobe AIR.

Książka „ActionScript 3.0 dla Adobe Flash CS4/CS4 PL Professional. Oficjalny podręcznik” została przygotowana w oparciu o oficjalny program szkoleniowy Adobe Systems Incorporated, opracowany przez ekspertów Adobe. Poszczególne lekcje zawierają podstawowe i zaawansowane informacje, a także liczne wskazówki oraz opisy technik czy mechanizmów, które umożliwiają uzyskanie wysokiej efektywności pracy. Z podręcznikiem nauczysz się sprawnie posługiwać tym wyjątkowym językiem programowania, tworzyć pliki kodu ActionScript, obsługiwać zdarzenia i funkcje. Dowiesz się, jak wzbogacać Twoje aplikacje o takie elementy, jak animacja, dźwięk, wideo i wyszukana grafika, aby zadowolić najbardziej wyrafinowane gusta użytkowników.

- Formaty plików Flash i ActionScript 3.0
- Oś czasu
- Zdarzenia i funkcje
- Animacja, dźwięk, wideo i ActionScript
- Tworzenie instalacji klasy we Flashu
- Ładowanie zawartości w czasie pracy aplikacji
- Narzędzia w środowisku testowania
- Tablice i pętle
- Lista odtwarzania w formacie XML
- Zaawansowane techniki animacji i pracy z grafiką
- Drukowanie i wysyłanie poczty

Opanuj ActionScript 3.0 i twórz efektowne animacje we Flashu CS4!

SPIS TREŚCI

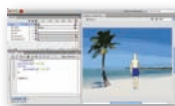
JAK ZACZAĆ

Seria „Oficjalny podręcznik”	9
Wymagane przygotowanie	9
Instalowanie Flasha	10
Konfiguracja sprzętowa komputera	10
Kopiowanie plików lekcji	10
Jak korzystać z tej książki	11
Konwencje typograficzne	11
Inne źródła informacji	12
Informacje na temat ActionScriptu	13
Aktualizacje Flasha	13
Wersja programu Flash Player	13
Użytkownicy wersji CS3	14
Certyfikaty Adobe	14

ACTIONSCRIPT 3.0 — WPROWADZENIE

Krótką historia Flasha i ActionScriptu	15
Początkujący programista języka ActionScript	16
Doświadczenie z ActionScript 1.0 i 2.0	16

1 OŚ CZASU



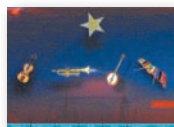
Przegląd lekcji	20
Pierwsze kroki	22
Wiązanie kodu z osią	23
Propozycje samodzielnych ćwiczeń	31

2 ZDARZENIA I FUNKCJE



Przegląd lekcji	34
Tworzenie detektorów zdarzeń i funkcji obsługi zdarzeń	36
Sterowanie odtwarzaniem kliknięciami przycisków	38
Propozycje samodzielnych ćwiczeń	48

3 ANIMACJA I ACTIONSCRIPT



Przegląd lekcji.....	50
Przykładowy projekt.....	52
Właściwości klipów.....	52
Obiekty typu Tween — animacje z klatkami pośrednimi w ActionScripte ..	59
Propozycje samodzielnych ćwiczeń	64

4 KOD ACTIONSCRIPT W PLIKACH ZEWNĘTRZNYCH



Przegląd lekcji.....	68
Tworzenie pliku kodu ActionScript	70
Tworzenie instancji klasy we Flashu	76
Propozycje samodzielnych ćwiczeń	83

5 ŁADOWANIE ZAWARTOŚCI W CZASIE PRACY APLIKACJI



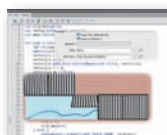
Przegląd lekcji.....	86
Tworzenie instancji składnika List i określanie jego parametrów	88
Dodawanie instancji składnika UILoader	90
Detektor zdarzenia CHANGE składnika List	91
Ładowanie plików SWF	91
Plik galerii	93
Dodawanie paska przewijania pola tekstowego	98
Propozycje samodzielnych ćwiczeń	99

6 STEROWANIE ŁADOWANIEM ZAWARTOŚCI



Przegląd lekcji.....	102
Narzędzia w środowisku testowania.....	104
Przygotowywanie pola tekstowego i paska postępu	108
Wstawianie kodu śledzącego pracę składnika UILoader	110
Wiązanie odtwarzania klatek z procesem ładowania	116
Propozycje samodzielnych ćwiczeń	123

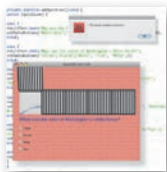
7 TABLICE I PĘTLE



Przegląd lekcji.....	126
Początkowa wersja przykładowego projektu	128
Dodawanie obiektów MovieClip z Biblioteki	130
Generowanie wielu obiektów przy użyciu pętli for	133
Detektory zdarzeń kliknięcia obiektów Block	137

Animacja inicjowana zdarzeniem ENTER_FRAME	139
Funkcja testDone()	143
Propozycje samodzielnych ćwiczeń	149

8 QUIZ Z PRZYCISKAMI OPCJI W PLIKU ACTIONSCRIPTU



Przegląd lekcji	152
Początkowa wersja przykładowego projektu	154
Tworzenie nowego pliku ActionScriptu	154
Podstawowe elementy klasy RadioButtonQuiz	155
Deklaracje zmiennych quizu	158
Funkcje quizu	160
Formatowanie tekstu	163
Tworzenie quizu	165
Sprawdzanie odpowiedzi — funkcja checkAnswer()	178
Włączanie nowej klasy do projektu	183
Propozycje samodzielnych ćwiczeń	186

9 DŹWIĘK I ACTIONSCRIPT



Przegląd lekcji	188
Początkowa wersja przykładowego projektu	190
Suwak — składnik Slider	192
Klasy Sound, SoundChannel i SoundTransform	193
Wstawianie tytułów piosenek — pętla for	197
Ukrywanie niepotrzebnych suwaków	199
Programowanie przycisków do wybierania utworów	200
Wyświetlanie suwaków	204
Detektor zdarzenia ID3 obiektu Sound	205
Obiekt TextFormat	208
Oprogramowanie suwaków	210
Propozycje samodzielnych ćwiczeń	212

10 LISTA ODTWARZANIA W FORMACIE XML



Przegląd lekcji	214
Ogólna struktura pliku XML	216
Początkowa wersja przykładowego projektu	219
Obiekt XML zamiast tablicy songList	220
Ładowanie zewnętrznej listy odtwarzania przy użyciu klasy URLLoader	221

Zdarzenia COMPLETE i IO_ERROR	222
Przenoszenie detektorów zdarzeń do funkcji xmlLoaded()	223
Aktualizacja funkcji chooseSong()	229
Hiperłącza korzystające z danych XML	232
Zmianie listy utworów	234
Propozycje samodzielnych ćwiczeń	236

11 WIDEO I ACTIONSCRIPT



Przegląd lekcji	238
Zawartość folderu lekcji	240
Składnik FLVPlayback	241
Konfigurowanie właściwości składnika FLVPlayback	242
Właściwości obiektu FLVPlayback w ActionScriptcie	245
Kolor	248
Składnik FLVCaptioning	252
Lista odtwarzania w formacie XML	255
Tryb pełnoekranowy	262
Propozycje samodzielnych ćwiczeń	265

12 ZAAWANSOWANE TECHNIKI ANIMACJI I PRACY Z GRAFIKĄ



Przegląd lekcji	268
Kinematyka odwrotna	270
Przegląd przykładowego projektu	272
Animacje IK w ActionScriptcie	274
Obiekty Sound i SoundChannel	281
Odtwarzanie i zatrzymywanie odtwarzania efektów dźwiękowych	282
Kamera i ActionScript	283
Klasy Bitmap i BitmapData	288
Pixel Bender Toolkit	291
Propozycje samodzielnych ćwiczeń	303

13 DRUKOWANIE I WYSYŁANIE POCZTY



Przegląd lekcji	308
Przegląd przykładowego projektu	310
Proste łącze do wysyłania poczty	311
Wysyłanie poczty z Flasha	312

Klasa PrintJob — funkcja drukowania w projekcie	317
Propozycje samodzielnych ćwiczeń	326

14 APLIKACJE ADOBE AIR



Przegląd lekcji	328
AIR i ActionScript	331
Ustawienia publikowania dla projektu AIR.....	332
Dołączanie pliku do aplikacji AIR	336
Tworzenie aplikacji AIR	337
Przegląd przykładowego projektu	339
Wykrywanie zdarzeń przeciągania obiektów do okna.....	341
Propozycje samodzielnych ćwiczeń	347
Skorowidz.....	349

5

ŁADOWANIE ZAWARTOŚCI W CZASIE PRACY APLIKACJI

Przegląd lekcji

W tej lekcji czytelnik dowie się, jak:

- pracować ze składnikami Flash CS4 User Interface;
- tworzyć obiekty List i określać ich parametry;
- aktywować detektor zdarzeń zmianą wyboru w obiekcie List;
- używać składnika UI Loader do sterowania ładowaniem i wyświetlaniem plików SWF i grafiki rastrowej;
- zmieniać plik źródłowy składnika UI Loader z poziomu ActionScriptu;
- używać klasy URL Loader do ładowania danych tekstowych z plików zewnętrznych;
- dodawać detektor zdarzeń reagujący na zakończenie ładowania danych;
- modyfikować właściwości pola tekstowego w ActionScriptie;
- używać składnika UI Scroll Bar do tworzenia pól tekstowych z mechanizmem przewijania zawartości.



Lekcja trwa około 2,5 godziny.

Jeżeli systematycznie zapoznawaliśmy się z materiałem poprzednich lekcji, znamy już pokazany zbiór metod pracy z ActionScriptem 3.0 i możemy wprowadzać do plików Flasha wiele różnorodnych efektów i funkcji. Jednak większość projektów we Flashu nie składa się z pojedynczego pliku, ale całej kolekcji plików SWF oraz plików z danymi i treścią multimedialną, których zawartość jest ładowana w czasie wykonania.



W tej lekcji utworzymy prostą galerię obrazków i włączymy ją do większego projektu we Flashu

Ponieważ jednym z głównych tematów tej lekcji jest integrowanie wielu plików w jeden projekt we Flashu, zasób materiałów na dysku CD jest znacznie bogatszy.

Zapoznajmy się z zawartością folderu *Lessons\Lesson05*. Znajdziemy w nim podfolder *Images* z plikami JPG i podfolder *Text* z plikami tekstowymi ASCII. Wszystkie te elementy włączymy do projektu we Flashu przy użyciu ActionScriptu.

W podfolderze *Start* znajduje się plik *lesson05_start fla*, główny plik, z którym będziemy pracować w tej lekcji. W tym samym folderze są również pliki *instruments.swf* i *paint.swf*. Są to finalne wersje projektów z lekcji 3. i 4. Lekcję rozpoczniemy od załadowania tych dwóch plików SWF do projektu *lesson05_start fla* przy użyciu instancji składnika *List* (innymi słowy, przy użyciu obiektu typu *List*). Następnie utworzymy nowy plik galerii, który pozwoli użytkownikowi wybierać elementy z listy miniatur i wyświetlać duże wersje reprezentowanych przez nie obrazków. Każdy obrazek będzie podpisany. Podpisy będą ładowane z odrębnych plików tekstowych. Gotowa galeria zostanie dodana do listy plików, które można załadować do projektu *lesson05_start fla*.

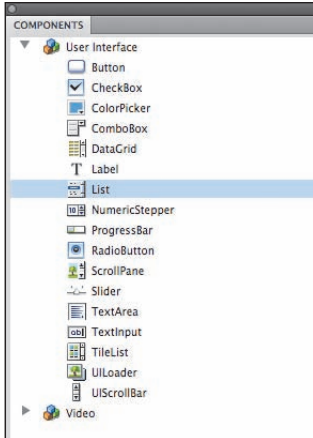
Tworzenie instancji składnika *List* i określanie jego parametrów

Składnik *List* (*lista*), jeden z wielu standardowych składników, które można włączać do projektów we Flashu CS4, pozwala w prosty sposób tworzyć listy obiektów do wyboru przez użytkownika. Konfigurację składnika określa grupa parametrów, których wartości można określać z poziomu interfejsu Flasha lub ActionScriptu. W ten sposób decydujemy o treści etykiet elementów listy i powiązaniach tych elementów z danymi. Składnik generuje również zdarzenia powiązane ze zmianami zaznaczenia na liście.

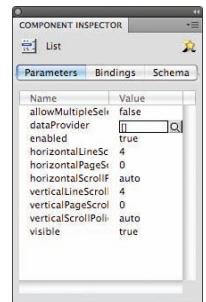
Rozpoczynamy lekcję od otwarcia pliku *lesson05_start fla* z folderu *Lessons\Lesson05\Start*. Zwróćmy uwagę, że jest to ten sam projekt, z którym pracowaliśmy w lekcji 1., „Oś czasu”, i w lekcji 2., „Zdarzenia i funkcje”. W tej lekcji przygotowujemy interfejs aplikacji.

- 1 Na osi czasu, nad warstwą *buttons*, dodajemy nową warstwę i nadajemy jej nazwę **components**.
- 2 Zaznaczamy klatkę 50 (o etykiecie *home*) nowej warstwy i dodajemy klatkę klawiszową (klawisz *F6*).
- 3 Otwieramy panel *Components* (*Składniki*) (polecenie *Window/Components* (*Okno/Składniki*)).

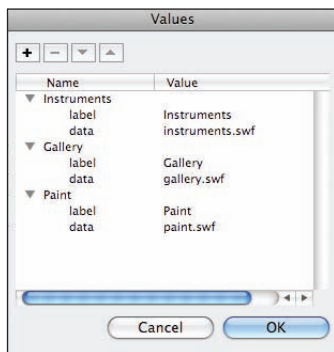
- 4 W panelu *Components* (*Składniki*) rozwijamy gałąź *User Interface* (*interfejs użytkownika*) i zaznaczamy składnik *List* (*lista*).



- 5 Zwracając uwagę na to, aby w dalszym ciągu zaznaczona była klatka 50 nowej warstwy components, przeciągamy zaznaczoną w panelu pozycję *List* na stół montażowy. Użyjemy tego składnika do utworzenia listy plików, które użytkownik może zaznaczyć i załadować do projektu.
- 6 Zaznaczamy nowy składnik typu *List* na stole montażowym i wyświetlamy panel *Properties* (*Właściwości*) (polecenie *Window/Properties* (*Okno/Właściwości*)).
- 7 W panelu właściwości nadajemy obiektowi nazwę instancji *loadList*.
- 8 Ponownie w panelu właściwości, zmieniamy wartość właściwości *X* obiektu *loadList* na **30**, a *Y* na **150**.
- 9 Zmieniamy wartości właściwości *W* (szerokość) i *H* (wysokość) na, odpowiednio, **140** i **60**.
- 10 Zwracając uwagę, aby w dalszym ciągu zaznaczony był ten sam obiekt *loadList*, otwieramy panel *Component Inspector* (*Inspektor składników*) (polecenie *Window/Component Inspector* (*Okno/Inspektor składników*)).
- 11 Zaznaczamy parametr *dataProvider* i klikamy obrazek lupy pojawiający się w prawej części pola tekstowego. Otwieramy w ten sposób okno, w którym można wprowadzać etykiety i określać powiązania elementów listy z danymi.
- 12 Dodajemy do listy trzy nowe elementy, klikając trzykrotnie przycisk **+**.
- 13 Zaznaczamy parametr *label* pierwszego elementu i w polu tekstowym po prawej stronie wprowadzamy **Instruments**.
Będzie to etykieta pierwszego elementu listy.
- 14 Zaznaczamy parametr *data* pierwszego elementu i wprowadzamy wartość **instruments.swf**.



Dane przypisane poszczególnym elementom posłużą do przechowywania nazw plików, które będą ładowane po wybraniu tych elementów.



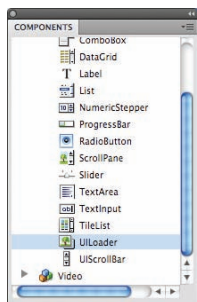
- 15 Dla drugiego elementu listy wprowadzamy etykietę **Gallery** i dane **gallery.swf**. Plik galerii utworzymy później (w tej lekcji).
- 16 Dla trzeciego elementu listy wprowadzamy etykietę **Paint** i dane **paint.swf**. Za chwilę dodamy kod, który będzie powodował, że zaznaczenie tego elementu na liście będzie prowadziło do załadowania końcowej wersji aplikacji do rysowania, którą przygotowaliśmy w lekcji 4., „Kod ActionScript w plikach zewnętrznych”.
- 17 Klikamy przycisk **OK**, aby zamknąć okno *Values (Wartości)*.

Dodawanie instancji składnika UILoader

W dalszej części tej lekcji nauczymy się ładować dane multimedialne do Flasha przy użyciu ActionScriptu. Jeżeli jednak chcemy załadować plik SWF, JPG, PNG lub GIF, użycie składnika UILoader pozwoli zaoszczędzić nieco pracy. Tutaj użyjemy go do załadowania plików SWF do projektu *lesson05_start fla*. Później w podobny sposób ładować będziemy do pliku galerii obrazki w formacie JPG. Przy ładowaniu plików tekstowych posłużymy się już ActionScriptem. Plików tekstowych nie można ładować przy użyciu składnika UILoader.

Zacniemy od umieszczenia obiektu UILoader na stole montażowym.

- 1 Po zaznaczeniu klatki 50 (home) warstwy components i wyświetleniu panelu *Components (Składniki)*, zaznaczamy składnik UILoader w grupie *User Interface (interfejs użytkownika)*.
- 2 Przeciągamy zaznaczony składnik na stół montażowy.
- 3 W panelu *Properties (Właściwości)* wprowadzamy nazwę instancji dla obiektu UILoader na stole montażowym: **loadWindow**.



- 4 Wprowadzamy parametry położenia i wielkości obiektu `loadWindow` (również w panelu *Properties*): $X = 200$, $Y = 135$, $W = 550$ i $H = 400$. Ładowane pliki SWF będą wyświetlane w rozmiarze 550 na 400 pikseli.

Detektor zdarzenia CHANGE składnika List

Gdy użytkownik wybierze jedną z pozycji wyświetlanych przez składnik List, nastąpi zdarzenie CHANGE. Funkcję obsługi tego zdarzenia definiujemy i uaktywniamy w podobny sposób jak dla zdarzeń opisywanych wcześniej.

- 1 Upewniamy się, że panele *Timeline (Oś czasu)* i *Actions (Operacje)* są widoczne, po czym zaznaczamy klatkę 50 (home) warstwy actions.
- 2 Umieszczamy kursor w wierszu pod wprowadzonym wcześniej kodem i wpisujemy:
`loadList.addEventListener(Event.CHANGE, loadFile);`

```
function loadFile(e:Event):void {  
  
}
```

Taka konstrukcja powinna wyglądać znajomo. Detektor zdarzenia CHANGE dodajemy w taki sam sposób jak detektory zdarzeń związanych z myszą i rozpoczęciem odtwarzania klatki.

Funkcja `loadFile()` jest teraz wywoływana za każdym razem, gdy użytkownik wybiera kliknięciem element listy. Naszym kolejnym zadaniem jest wpisanie kodu tej funkcji. Będzie on ładował odpowiedni plik SWF wyświetlany przez instancję składnika `UILoader`.

Ładowanie plików SWF

Załadowanie pliku SWF, JPG, PNG lub GIF do składnika `UILoader` sprowadza się do przypisania właściwości `source` tego składnika nazwy pliku. Składnia polecenia jest następująca:

```
UILoaderInstanceName.source = "Ścieżka pliku";
```

Aby na przykład załadować plik *instruments.swf* do instancji składnika o nazwie `loadWindow`, piszemy:

```
loadWindow.source = "instruments.swf";
```

W naszym projekcie potrzebna jest funkcja, która ładuje plik określony danymi powiązаныmi z elementem listy. Nazwy plików wprowadziliśmy wcześniej jako parametry `dataProvider`. Informacja ta zostanie wykorzystana w kodzie funkcji. W efekcie jeżeli użytkownik wybierze na przykład element o etykiecie *Paint*, do obiektu `UILoader`

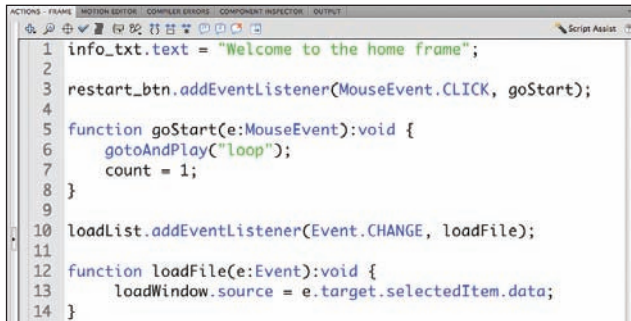
zostanie załadowany plik *paint.swf*. Ciąg „*paint.swf*” zostanie odczytany z parametru *dataProvider* wybranego elementu listy.

- 1 W nowo utworzonej funkcji *loadFile()* wprowadzamy polecenie przypisania danych do właściwości *source*:

```
function loadFile(e:Event):void {  
    loadWindow.source = e.target.selectedItem.data;  
}
```

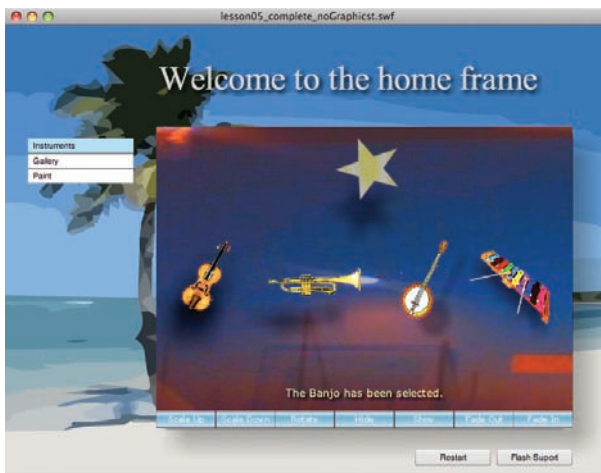
e to obiekt *Event* (zdarzenie), *e.target* to lista, właściwość *selectedItem* to wybrany przez użytkownika element, a właściwość *data* to dane wprowadzone w inspektorze składników.

Całość kodu w klatce 50 powinna teraz wyglądać następująco:



```
1 info_txt.text = "Welcome to the home frame";  
2  
3 restart_btn.addEventListener(MouseEvent.CLICK, goStart);  
4  
5 function goStart(e:MouseEvent):void {  
6     gotoAndPlay("loop");  
7     count = 1;  
8 }  
9  
10 loadList.addEventListener(Event.CHANGE, loadFile);  
11  
12 function loadFile(e:Event):void {  
13     loadWindow.source = e.target.selectedItem.data;  
14 }
```

- 2 Zapisujemy i testujemy projekt.
- 3 W środowisku testowania zaznaczamy na liście element *Paint*. Spowoduje to załadowanie do interfejsu pliku *paint.swf*.
- 4 Zaznaczamy na liście element *Instruments*. Zostanie załadowany plik *instruments.swf*.



- 5 Zaznaczamy na liście element *Gallery*. To spowoduje wygenerowanie błędu, ponieważ plik *gallery.swf* nie został jeszcze utworzony. Zajmiemy się tym później. W lekcji 13., „Drukowanie i wysyłanie poczty”, wrócimy do tematu obsługi pojawiających się w czasie wykonania błędów. Będzie to niezbędne, aby uchronić użytkownika przed niezrozumiałymi komunikatami w przypadku zakłóceń przy ładowaniu plików.

Plik galerii

Teraz przejdziemy do przygotowywania pliku galerii, do którego odwołuje się składnik List. Galeria będzie aplikacją, która pozwala użytkownikowi wybierać obrazki JPG ładowane przez obiekt UILoader przy użyciu miniatur. Zaznaczenie miniatury będzie dodatkowo powodowało załadowanie tekstu z pliku zewnętrznego i przypisanie pobranego ciągu znaków do pola tekstowego na stole montażowym. Wstępna wersja projektu to plik *gallery fla* w folderze *Lesson05/Start*. Dodamy do niego kod ActionScript realizujący wszystkie potrzebne funkcje. Rozpoczniemy jednak od zapoznania się z elementami, które zostały już dla wygody czytelnika przygotowane.

Plik *gallery fla*

W pliku przygotowane są elementy graficzne i ogólny układ galerii. Naszym zadaniem będzie uzupełnienie projektu o odpowiedni kod ActionScript, czyli polecenia odpowiedzialne za ładowanie tekstu i obrazków.

- 1 Otwieramy plik *gallery fla* z folderu *Lessons/Lesson05/Start*.
Na osi czasu widać cztery warstwy, a na stole montażowym znajdują się trzy obiekty. Nie zostały jeszcze zdefiniowane żadne operacje. Naszym podstawowym zajęciem w tej lekcji będzie wprowadzanie kodu w warstwie *actions*. W warstwie *loader* znajduje się obiekt UILoader.
- 2 Zaznaczamy instancję składnika UILoader. W panelu właściwości widać, że jej nazwa to *ldr*.
- 3 Zaznaczamy dynamiczne pole tekstowe w warstwie *text*. Nazwa instancji to *info*.
- 4 Zaznaczamy w warstwie *thumbs* klip zawierający zbiór miniatur. Jego nazwa to *thumbs_mc*.
- 5 Dwukrotnie klikamy klip *thumbs_mc*.
Teraz widać, że klip składa się z siedmiu przycisków z obrazkami. Gdy zaznaczymy kolejne przyciski, widzimy zmiany nazwy instancji w panelu *Properties* (*Właściwości*) — od *btn1* do *btn7*. Ponieważ przyciski znajdują się wewnątrz klipu *thumbs_mc*, ścieżka odwołań do nich będzie miała postać *thumbs_mc.btn1*, *thumbs_mc.btn2* itd.
- 6 Wracamy do osi czasu, wybierając polecenie menu *Edit/Edit document* (*Edytuj/Edytuj dokument*).

Detektory zdarzeń przycisków

We wcześniejszych lekcjach używaliśmy już metody `addEventListener()` do wskazania funkcji wykonywanej po kliknięciu przycisku. Powtórzymy teraz taką samą procedurę dla przycisków wewnątrz klipu `thumbs_mc`. Teraz jednak będziemy musieli zadbać o odpowiednie ścieżki wskazujące przyciski — są one nieco bardziej rozbudowane.

1 Po zaznaczeniu klatki 1 warstwy `actions` wstawiamy kursor tekstowy w pierwszym wierszu panelu *Actions (Operacje)*.

2 Pamiętając o odpowiedniej konstrukcji ścieżki odwołania do obiektu przycisku, wprowadzamy kod wywołań metody `addListener`:

```
thumbs_mc.btn1.addEventListener(MouseEvent.CLICK, ldr1);
thumbs_mc.btn2.addEventListener(MouseEvent.CLICK, ldr2);
thumbs_mc.btn3.addEventListener(MouseEvent.CLICK, ldr3);
thumbs_mc.btn4.addEventListener(MouseEvent.CLICK, ldr4);
thumbs_mc.btn5.addEventListener(MouseEvent.CLICK, ldr5);
thumbs_mc.btn6.addEventListener(MouseEvent.CLICK, ldr6);
thumbs_mc.btn7.addEventListener(MouseEvent.CLICK, ldr7);
```

Kliknięcia przycisków będą teraz powodować wywołania funkcji `ldr1`, `ldr2` itd. Naszym kolejnym zadaniem jest zdefiniowanie tych funkcji.

3 W wierszu pod wywołaniami metody `addListener` wpisujemy definicję funkcji `ldr1()`:

```
function ldr1(e:Event) {
    ldr.source = "../images/image1.jpg";
}
```

Po kliknięciu pierwszego przycisku funkcja ładuje obrazek *image1.jpg* do obiektu `UILoader` o nazwie `ldr`. Zwróćmy uwagę na zapis ścieżki do pliku JPG. Fragment `../` nakazuje przejść o jeden folder wyżej w hierarchii folderów na dysku. Punktem odniesienia jest lokalizacja pliku Flasha galerii. Tam wyszukiwany jest folder *images*, a w nim plik *image1.jpg*. Jeżeli taki zapis jest dla czytelnika nowością, pomocne będzie wyświetlenie drzewa podfolderów *Lessons\Lesson05* i porównanie lokalizacji pliku *gallery.swf* i folderu *images*.

4 Uzupełniamy funkcję o dodatkowy wiersz:

```
function ldr1(e:Event) {
    ldr.source = "../images/image1.jpg";
    textLoad("../text/picture1.txt", 0xAAFFAA);
}
```

Kliknięcie przycisku powoduje załadowanie obrazka przez obiekt `UILoader`. Dodany wiersz wywołuje funkcję `textLoad()`, której zadaniem jest ładowanie zawartości plików tekstowych do pola tekstowego. Funkcja ta jeszcze nie istnieje. Jeżeli przed jej wpisaniem podejmiemy próbę testowania projektu, zostanie wyświetlony komunikat błędu. Zwróćmy uwagę, że funkcja pobiera dwa parametry. Pierwszy to ścieżka pliku tekstowego, a drugi to numer koloru, który zostanie użyty dla tła pola tekstowego. Funkcję `textLoad()` utworzymy już wkrótce, ale najpierw dodamy funkcje dla pozostałych przycisków.

5 Przygotowujemy analogiczne funkcje dla pozostałych sześciu przycisków.

Zwróćmy uwagę, że we wcześniejszych lekcjach po każdym wywołaniu metody `addEventListener()` od razu wprowadzaliśmy odpowiednią funkcję obsługi zdarzenia. Tutaj wywołania metody dodającej detektor zdarzeń są zebrane razem, a dopiero po wszystkich następują definicje kolejnych funkcji. Kolejność może być dowolna.

Po wpisaniu całego kodu panel *Actions (Operacje)* powinien wyglądać podobnie do przedstawionego na rysunku:



```
1 thumbs_mc.btn1.addEventListener(MouseEvent.CLICK, ldr1);
2 thumbs_mc.btn2.addEventListener(MouseEvent.CLICK, ldr2);
3 thumbs_mc.btn3.addEventListener(MouseEvent.CLICK, ldr3);
4 thumbs_mc.btn4.addEventListener(MouseEvent.CLICK, ldr4);
5 thumbs_mc.btn5.addEventListener(MouseEvent.CLICK, ldr5);
6 thumbs_mc.btn6.addEventListener(MouseEvent.CLICK, ldr6);
7 thumbs_mc.btn7.addEventListener(MouseEvent.CLICK, ldr7);
8
9 function ldr1(e:Event) {
10     ldr.source = "../images/image1.jpg";
11     textLoad("../text/picture1.txt", 0xAFFFAA);
12 }
13 function ldr2(e:Event) {
14     ldr.source = "../images/image2.jpg";
15     textLoad("../text/picture2.txt", 0xCCCEE);
16 }
17 function ldr3(e:Event) {
18     ldr.source = "../images/image3.jpg";
19     textLoad("../text/picture3.txt", 0xCCEEC);
20 }
21 function ldr4(e:Event) {
22     ldr.source = "../images/image4.jpg";
23     textLoad("../text/picture4.txt", 0xFFCCCC);
24 }
25 function ldr5(e:Event) {
26     ldr.source = "../images/image5.jpg";
27     textLoad("../text/picture5.txt", 0xCCDDAA);
28 }
29 function ldr6(e:Event) {
30     ldr.source = "../images/image6.jpg";
31     textLoad("../text/picture6.txt", 0xBBFFCC);
32 }
33 function ldr7(e:Event) {
34     ldr.source = "../images/image7.jpg";
35     textLoad("../text/picture7.txt", 0xCDEFED);
36 }
37
```

Ładowanie tekstu z pliku zewnętrznego

Teraz wpiszemy kod ładujący różne pliki tekstowe do pola tekstowego `info`. Składnik `UI Loader` używany do ładowania plików SWF i plików obrazków używa klasy `ActionScript Loader`. Użycie składnika z grupy `User Interface` pozwoliło nam uniknąć wpisywania kodu aplikacji ładującego obrazki — za całą operację odpowiada `UI Loader`. Ładowanie tekstu lub innych danych odbywa się nieco inaczej. Używamy w tym celu klasy o nazwie `URL Loader`. Ponieważ nie pomoże nam żaden standardowy składnik (*Component*) Flasha, musimy napisać kod tworzący nowy obiekt `URL Loader`.

- 1 Pod wpisaniem wcześniej w panelu *Actions (Operacje)* kodem wprowadzamy polecenie utworzenia nowego obiektu:

```
var loader:URLLoader = new URLLoader();
```

Kolejnym elementem jest funkcja `textLoad()`, która wykona operację ładowania tekstu z pliku zewnętrznego. Wywołanie tej funkcji występuje w każdej funkcji obsługi zdarzenia kliknięcia przycisku.

- 2 Pod wpisaniem wcześniej w panelu *Actions (Operacje)* kodem wprowadzamy:

```
function textLoad(file:String, color:uint) {  
    loader.load(new URLRequest(file));  
    info.backgroundColor = color;  
}
```

Funkcja `textLoad()` robi dwie rzeczy. Najpierw wywołuje metodę `load` klasy `URLLoader`. Metoda ta ładuje plik tekstowy. Pamiętajmy, że wywołanie tej funkcji następuje przy kliknięciu jednego z siedmiu przycisków. Gdy powrócimy do funkcji obsługi zdarzeń, zobaczymy w nich, że funkcji przekazywane są dwa parametry. Pierwszym jest ciąg znaków opisujący ścieżkę pliku, a drugi to numer koloru. Nazwy parametrów to `file` i `color`.

Parametr `file` służy do wskazywania ścieżki pliku, który zostanie załadowany metodą `load()`. Parametr `color` służy do określania koloru tła umieszczonego wcześniej na stole montażowym pola tekstowego o nazwie `info`.

Gdy funkcja zostaje wywołana, pierwszy wiersz ładuje tekst, a drugi zmienia kolor tła. Jednak w tym momencie tekst nie zostaje jeszcze wyświetlony. Wymaga to dodatkowo przypisania załadowanych danych do właściwości `text` pola tekstowego.

Wyświetlanie danych załadowanych z zewnętrznych plików tekstowych w polu tekstowym jest prostą operacją. Zanim jednak do niej przejdziemy, powinniśmy zadbać o bardzo istotną rzecz — sprawdzić, czy czynność pobierania danych z zewnątrz zakończyła się sukcesem.

Zdarzenie COMPLETE — sprawdzanie, czy tekst został załadowany

Zdarzenie `COMPLETE` obiektu `URLLoader` może zostać powiązane z funkcją obsługi zdarzenia (podobnie jak każde inne zdarzenie). Dodamy teraz odpowiedni detektor — ponownie użyjemy metody `addEventListener`.

- 1 W kolejnym wierszu panelu *Actions (Operacje)* wprowadzamy:

```
loader.addEventListener(Event.COMPLETE, displayText);  
function displayText(e:Event) {  
    info.text = (loader.data);  
}
```

Gdy obiekt `loader` zakończy ładowanie pliku tekstowego, funkcja obsługi zdarzenia `COMPLETE` wyświetli tekst w polu na stole montażowym.

Teraz możemy poświęcić nieco uwagi formatowaniu pola tekstowego. Będzie ono miało wyróżniające się tło i obramowanie, ustalimy też kolor obramowania.

Sprawdzanie wyniku operacji ładowania danych zewnętrznych

Do wyświetlenia załadowanego przez obiekt `URLLoader` tekstu wystarczy jeden wiersz:

```
info.text = (loader.data);
```

Całkiem naturalne byłoby dołączenie tego wiersza do funkcji `textLoad()`:

```
function textLoad(file:String, color:uint) {
    loader.load(new URLRequest(file));
    info.backgroundColor = color;
    info.text = (loader.data);
}
```

Choć takie rozwiązanie sprawdzałoby się zapewne w aplikacji uruchamianej lokalnie, może powodować wiele problemów, gdy plik są pobierane z serwera. Pamiętajmy, że wykonanie wiersza kodu w języku `ActionScript` zajmuje zazwyczaj niewielki ułamek sekundy. Gdy w jednym poleceniu nakazujemy załadowanie pliku tekstowego z dysku, który może znajdować się na innym komputerze, a dwa wiersze dalej umieszczamy instrukcję wyświetlenia pobranych danych, możemy być pewni, że wcześniej czy później opóźnienie w komunikacji między komputerami spowoduje, że pobierane dane nie będą gotowe do użycia. Wówczas zostanie wygenerowany komunikat błędu czasu wykonania.

Jest to problem, z którym należy liczyć się przy każdej operacji pobierania danych z lokalizacji zdalnej i który sprawia, że jedną z podstawowych reguł programowania jest potwierdzanie wyniku takiej operacji przed próbą użycia danych.

W `ActionScript`ie nie jest to trudne — zarówno klasa `Loader`, jak i `URLLoader` dysponują standardowym zdarzeniem `COMPLETE`, generowanym, gdy realizacja żądania pobrania danych kończy się sukcesem.

- 2 Dodajemy trzy wiersze przypisujące właściwości decydujące o formatowaniu:

```
function displayText(e:Event) {
    info.text = (loader.data);
    info.background = true;
    info.border = true;
    info.borderColor = 0x333333;
}
```

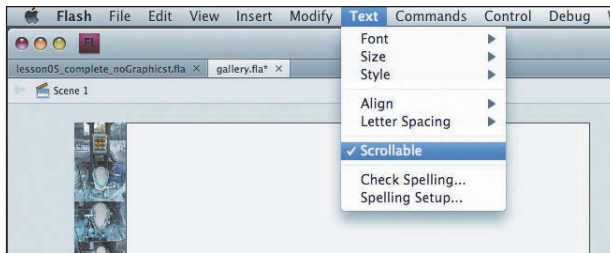
● **Uwaga:**
Formatowaniem tekstu zajmujemy się w lekcji 8, „Quiz z przyciskami opcji w pliku `ActionScriptu`”.

Dodawanie paska przewijania pola tekstowego

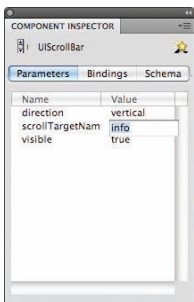
Zawartość ładowanych plików tekstowych nie mieści się w polu tekstowym na stole montażowym. Szczęśliwie się jednak składa, że Flash został wyposażony w składnik UIScrollBar, który pozwala wyposażyć pole w pasek przewijania.

Przewijanie tekstu to ważna funkcja, zwłaszcza w interfejsach, które dysponują niewielką ilością miejsca do wyświetlania informacji. Dodanie do pola tekstowego paska przewijania nie jest trudne. Naszym kolejnym zadaniem będzie powiązanie pola tekstowego `info` ze składnikiem UIScrollBar.

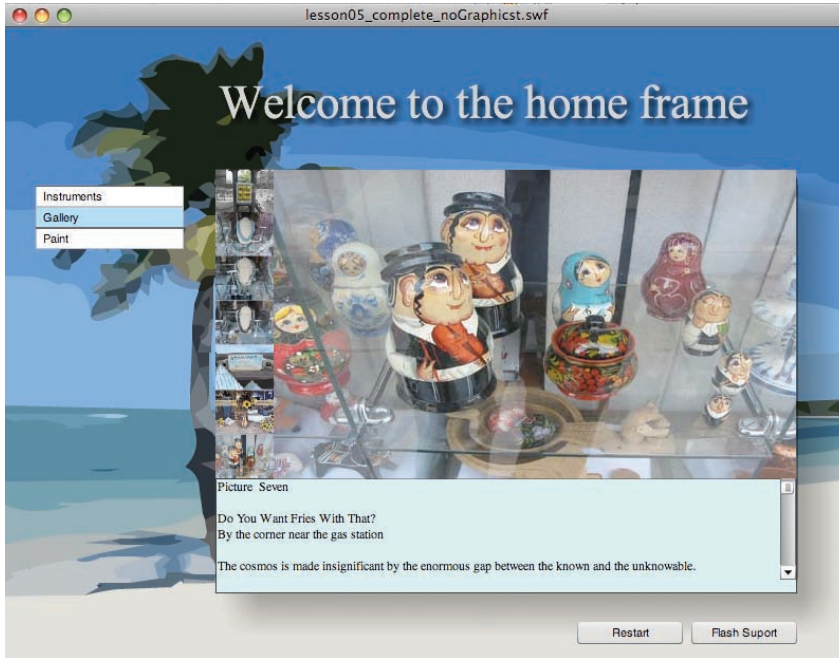
- 1 Zaznacz na stole montażowym pole tekstowe `info`.
- 2 Wybierz z menu *Text (Tekst)* polecenie *Scrollable (Przewijalny)*.



- 3 Poleceniem *Window/Components (Okno/Składniki)* otwieramy panel *Components (Składniki)* i zaznaczamy składnik UIScrollBar z grupy User Interface.
- 4 Przeciągamy składnik z panelu *Components (Składniki)* na stół montażowy, do prawego górnego rogu pola tekstowego `info`.
- 5 Po przeciągnięciu składnika UIScrollBar na stół montażowy otwieramy panel *Component Inspector (Inspektor składników)* (polecenie *Window/Component Inspector (Okno/Inspektor składników)*).
- 6 Zaznaczamy parametr `scrollTargetName`.
- 7 Wprowadzamy nazwę `info`.
- 8 Zapisujemy i testujemy projekt. Kliknięcie jednego z przycisków z miniaturami powoduje załadowanie obrazka i wyświetlenie go przez obiekt UI Loader oraz wyświetlenie tekstu w polu `info`. Dodatkowo zmieniany jest kolor tła pola tekstowego. W polu `info` można korzystać z paska przewijania.
- 9 Zapisujemy plik i powracamy do projektu `lesson05_start fla`.



- 10 Ponownie testujemy plik *lesson05_start fla*. Wybranie z listy elementu *Gallery* powoduje teraz otwieranie w obiekcie UI Loader nowego pliku galerii. Przyciski galerii powinny działać tak jak przy ostatniej próbie z plikiem *gallery fla*.



Propozycje samodzielnych ćwiczeń

Samodzielne próby dalszego rozbudowywania projektu pomogą ugruntować świeżo zdobytą wiedzę. Oto kilka propozycji:

- Utwórz nowy projekt we Flashu i dodaj go do listy w *lesson05_start fla* tak, aby stał się kolejnym elementem listy obiektów do wyświetlenia.
- Zastąp pliki JPG używane w projekcie *gallery fla* innymi obrazkami. Spróbuj tak zmodyfikować kod, aby pliki były pobierane z innej ścieżki.
- Spróbuj dołączyć do projektu inne składniki z grupy User Interface. Informacje o ich parametrach można znaleźć w Pomocy Flasha.

W następnej lekcji nauczymy się konstruować mechanizm ładowania wstępnego i monitorować proces pobierania zewnętrznych plików Flasha.

Pytania kontrolne

- 1 Jakie typy plików można załadować do projektu przy użyciu składnika UILoader?
- 2 Jakie zdarzenie składnika List jest generowane, gdy użytkownik wybiera elementy z listy?
- 3 Jakie zdarzenie klasy URLLoader jest generowane w chwili zakończenia pobierania danych?

Odpowiedzi

- 1 Składnik UILoader pozwala ładować pliki SWF, JPG, PNG i GIF.
- 2 Do wykrywania wyboru elementu z listy wyświetlanej przez składnik List służy zdarzenie CHANGE tego składnika.
- 3 Poprawne zakończenie ładowania danych przez obiekt URLLoader sygnalizuje zdarzenie COMPLETE.