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

Flash CS3 Professional PL. Techniki zaawansowane. Klatka po klatce

Autor: Russell Chun
Tłumaczenie: Joanna Pasek
ISBN: 978-83-246-1342-7
Tytuł oryginału: Flash CS3 Professional Advanced for Windows and Macintosh: Visual QuickPro Guide
Format: 168x237, stron: 530

Odkryj najgłębiej strzeżone tajemnice Flasha

- Jak tworzyć rozbudowane animacje?
- W jaki sposób kontrolować obiekty, wykorzystując ActionScript?
- Jak pobierać dane z zewnętrznych źródeł?

Flash CS3 Professional to kolejne wcielenie narzędzia, które zyskało uznanie dziesiątek tysięcy twórców witryn WWW. Jednak jego niesamowite możliwości w zakresie tworzenia grafiki i animacji to tylko wierzchołek góry lodowej. Swoją prawdziwą potęgę Flash ujawnia, gdy sięgniemy po rozwiązania oparte na wbudowanym, obiektowym języku programowania noszącym nazwę ActionScript 3.0. Za jego pomocą możemy kontrolować niemal każdy aspekt animacji i wszystkie obiekty wchodzące w jej skład, sterować wyświetlaniem i pobieraniem danych oraz odtwarzaniem dźwięku. Jesteśmy w stanie wykreować niesamowite efekty, jakich nigdy nie uzyskalibyśmy, wykorzystując inne techniki. W książce „Flash CS Professional PL. Techniki zaawansowane. Klatka po klatce" opisano techniki tworzenia animacji związane ze stosowaniem języka ActionScript. Czytając ją, poznasz podstawy tego języka, a także dowiesz się, w jaki sposób pisać skrypty i do jakich obiektów je dołączać. Nauczysz się kontrolować elementy graficzne, dźwiękowe, tekstowe i nawigacyjne umieszczone w prezentacji za pomocą poleceń ActionScriptu. Opanujesz sposoby łączenia witryn WWW tworzonych we Flashu z zewnętrznymi plikami, przeglądarką internetową i serwerem, na którym są publikowane. Znajdziesz tu także informacje o wykrywaniu błędów w skryptach i ich usuwaniu.

- Animacje poklatkowe i automatyczne
- Animowane maski
- Rotoskopia
- Edycja kodu ActionScript w panelu Actions
- Tworzenie funkcji
- Obsługa zdarzeń w animacji
- Zarządzanie listwami czasowymi
- Tworzenie elementów nawigacyjnych
- Pobieranie plików multimedialnych z zewnętrznych źródeł
- Przetwarzanie elementów graficznych i dźwiękowych oraz danych tekstowych
- Operacje matematyczne we Flashu
- Optymalizacja filmów

Odkryj zapierające dech w piersiach możliwości synergii Flasha i języka ActionScript!

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

Spis treści

Wstęp 9

Część I **Zaawansowane animacje** 13

Rozdział 1. **Tworzenie złożonych projektów** 15

Automatyczna animacja ruchu ... 16
Automatyczna animacja kształtu ... 28
Efekty specjalne ... 33
Animowane i złożone maski ... 38

Rozdział 2. **Praca z wideo** 49

Osadzanie plików wideo we Flashu ... 50
Zewnętrzne pliki wideo; stopniowe pobieranie filmów w trakcie odtwarzania ... 59
Rotoskopia ... 65

Część II **Interaktywność** 67

Rozdział 3. **Posługiwanie się językiem ActionScript** 69

Co kryje nazwa ActionScript 3.0? ... 70
Obiekty i klasy ... 71
Metody i właściwości ... 72
Używanie notacji kropkowej ... 73
Więcej na temat interpunkcji ... 75
Panel Operacje ... 76
Edycja kodu ActionScript ... 85
Używanie obiektów ... 88
Funkcje ... 97
Stosowanie komentarzy ... 101

Rozdział 4. **Więcej o przyciskach i obsłudze zdarzeń** 103

Odbieranie zdarzeń ... 104
Obsługa myszy ... 106
Klasa SimpleButton ... 110
Niewidoczne przyciski ... 114
Animowane przyciski a klipy filmowe ... 116
Złożone przyciski ... 119
Możliwości śledzenia przycisków ... 123
Zmiana zachowania przycisku ... 126

Dynamiczne tworzenie przycisków ... 129
Obsługa klawiatury ... 131
Menu podręczne ... 134
Tworzenie powtarzających się akcji ... 140
Obiekt Timer ... 141
Podsumowanie zdarzeń ... 144

Rozdział 5. **Praca z wieloma listwami czasowymi** **145**
Nawigowanie listwami czasowymi klipów filmowych ... 146
Ścieżki adresowe ... 147
Bezwzględne i względne ścieżki adresowe ... 151
Używanie akcji with do adresowania klipów filmowych ... 153
Klipy filmowe jako kontenery ... 155
Etykiety ujęć ... 159

Rozdział 6. **Zarządzanie komunikacją zewnętrzną** **163**
Komunikowanie się przez przeglądarkę internetową ... 164
Ładowanie zewnętrznych filmów Flasha ... 180
Kontrolowanie załadowanych filmów Flasha ... 185
Ładowanie zewnętrznych obrazów ... 190
Komunikacja z zewnętrznym wideo ... 193
Stosowanie projektorów i akcji fscommand ... 200
Komunikowanie się z drukarką ... 203
Wykrywanie pobierania danych z sieci — wskaźniki postępu ładowania ... 212

Część III **Sterowanie grafiką i dźwiękiem** **221**

Rozdział 7. **Sterowanie grafiką** **223**
Lista wyświetlania ... 224
Zmiana własności obiektu graficznego ... 225
Skalowanie i zniekształcanie obiektów ... 232
Zmiana koloru ... 236
Mieszanie kolorów ... 242
Użycie filtrów do tworzenia efektów specjalnych ... 245
Przeciąganie i upuszczanie obiektów ... 248
Wykrywanie kolizji obiektów ... 252
Dynamiczne tworzenie obiektów ... 255
Kontrolowanie kolejności nakładania obiektów ... 258
Dynamiczne tworzenie kształtów ... 261
Dynamiczne maski ... 278
Dostosowanie wyglądu wskaźnika myszy ... 284
Podstawy animacji za pomocą języka ActionScript ... 286
Obrazy bitmapowe ... 291
Tworzenie bitmap i dostęp do nich ... 292

Edycja bitmap ... 298
Używanie filtrów do modyfikacji bitmap ... 308
Animowanie bitmap ... 311

Rozdział 8. **Sterowanie dźwiękiem** **315**
Wprowadzanie dźwięków ... 316
Odtwarzanie dźwięków z biblioteki ... 317
Wczytywanie i odtwarzanie zewnętrznych dźwięków ... 319
Sterowanie odtwarzaniem dźwięków ... 321
Modyfikacja natężenia i balansu dźwięku ... 325
Wykrywanie zdarzeń dźwięku ... 330
Wykorzystanie znaczników informacyjnych plików MP3 ... 332
Wizualizacja dźwięku ... 335
Tworzenie dynamicznego sterowania dźwiękiem ... 338

Część IV **Przetwarzanie informacji** **343**

Rozdział 9. **Sterowanie przepływem informacji** **345**
Zmienne i wyrażenia ... 346
Wczytywanie zewnętrznych zmiennych ... 350
Przechowywanie i współdzielenie informacji ... 359
Modyfikacja zmiennych ... 365
Dynamiczne generowanie nazw zmiennych za pomocą wyrażeń ... 367
Testowanie informacji za pomocą instrukcji warunkowych ... 369
Alternatywa dla instrukcji warunkowej ... 374
Rozgałęzione instrukcje warunkowe ... 376
Łączenie warunków z operatorami logicznymi ... 380
Pętle ... 382

Rozdział 10. **Tekst** **387**
Wejściowe pole tekstowe ... 388
Dynamiczne pole tekstowe ... 390
Opcje pól tekstowych ... 392
Wyświetlanie tekstu w formacie HTML ... 395
Właściwości klasy TextField ... 400
Dynamiczne generowanie pól tekstowych ... 410
Edycja tekstu w polach tekstowych ... 412
Formatowanie pól tekstowych z wykorzystaniem zewnętrznych arkuszy stylów ... 422
Wykrywanie aktywnego pola tekstowego ... 426
Uaktywnianie pól tekstowych i zaznaczanie tekstu w polach tekstowych ... 429
Analiza tekstu ... 431
Aranżacja tekstów ... 441

Rozdział 11. **Przetwarzanie informacji** **445**
Obliczenia przy użyciu klasy Math ... 446
Wyliczanie kątów ... 447
Tworzenie ukierunkowanego ruchu ... 456
Obliczanie odległości ... 460
Generowanie liczb losowych ... 462
Przechowywanie informacji w tablicach ... 463
Obsługa obiektów umieszczonych w tablicy ... 470
Data i czas ... 475

Rozdział 12. **Zarządzanie zawartością i usuwanie błędów** **483**
Współdzielenie symboli z biblioteki ... 484
Współdzielenie czcionek ... 491
Dołączanie zewnętrznego kodu ActionScript ... 494
Panel Eksplorator filmu ... 497
Śledzenie zmiennych w panelu Wyjście ... 501
Określanie typu zmiennej ... 503
Optymalizacja filmu ... 504
Unikanie typowych błędów ... 508

Dodatek A **Klawisze i odpowiadające im kody** **509**

Skorowidz **511**

Sterowanie dźwiękiem 8

Wprowadzenie dźwięku do filmu Flasha wzbogaca dowolną animację lub nawet najprostszy projekt interaktywny, ponieważ angażuje się więcej zmysłów użytkownika. Muzyka odtwarzana w tle wprowadza w odpowiedni nastrój, narracja ułatwia opowiedzenie historyjki, a dźwięki odtwarzanie na przykład po kliknięciu przycisku lub przy przeciąganiu obiektów po scenie dostarczają dodatkowych informacji. Flash obsługuje kilka formatów zapisu dźwięku, przede wszystkim formaty WAV, AIF i MP3. Pozwalają one korzystać z bardzo szerokiego spektrum dźwięków. Dodatkowo Flash potrafi dynamicznie ładować zewnętrzne pliki MP3 w trakcie odtwarzania filmu, co pozwala bez trudu tworzyć filmy zawierające dużą ilość dźwięku.

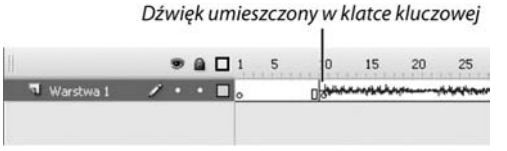
W tym rozdziale zajmiemy się dźwiękiem i klasami, które są przeznaczone do jego obsługi: klasą `Sound`, `SoundChannel`, `SoundMixer`, `SoundTransform` oraz `SoundEvent`. Dowiesz się, jak przy użyciu klasy `Sound` odtwarzać dźwięki z biblioteki bez potrzeby umieszczania ich w klatkach kluczowych. Nauczysz się obsługiwać dźwięki znajdujące się w zewnętrznych plikach, a tym samym wydajnie zarządzać filmami Flasha i odtwarzaną w nich muzyką. Z poziomu kodu ActionScript można rozpocząć odtwarzanie dźwięku, zatrzymać je, zmienić głośność lub balans. W ten sposób można dynamicznie reagować na działania użytkownika lub animację. Dowiesz się również, w jaki sposób wykorzystać właściwości i zdarzenia klasy `Sound`, by zapewnić synchronizację dźwięku z animacją lub z innymi dźwiękami.

Wszystkie te narzędzia służą do tego, by zapewnić elastyczny i skuteczny system integracji dźwięku z filmami Flasha. Można na przykład wykonać suwak zapewniający zmianę głośności lub uzależnić odtwarzaną muzykę od aktualnego przebiegu gry. Można wykonać pokaz slajdów zsynchronizowany z muzyką, a nawet własny odtwarzacz plików MP3.

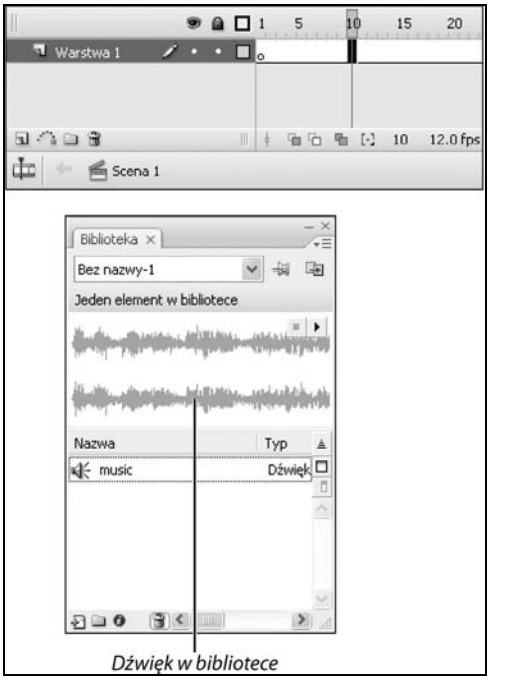
Wprowadzanie dźwięków

Jest kilka sposobów dołączania dźwięków do filmu Flasha. Najprostszym z nich jest zaimportowanie dźwięku i ręczne umieszczenie go w klatce kluczowej. Na listwie czasowej wyświetlana jest wtedy graficzna reprezentacja fali dźwiękowej, która pomaga się nam zorientować, kiedy dźwięk zaczyna być odtwarzany, a kiedy się kończy (rysunek 8.1). Drugi sposób to zaimportować dźwięk i odtworzyć go dynamicznie w trakcie trwania filmu. Dopóki nie zarządzisz odtworzenia takiego dźwięku za pomocą ActionScript, pozostaje on w bibliotece (rysunek 8.2). Trzeci sposób polega na dynamicznym załadowaniu i odtworzeniu dźwięku zapisanego w pliku zewnętrznym (rysunek 8.3). Ten rozdział jest poświęcony drugiej i trzeciej z wymienionych metod. Odtwarzając dźwięk dynamicznie, możemy sterować momentem jego pojawienia się i głośnością (także oddzielnie dla lewego i prawego głośnika) oraz pobierać informacje na temat postępu ładowania lub postępu odtwarzania plików dźwiękowych.

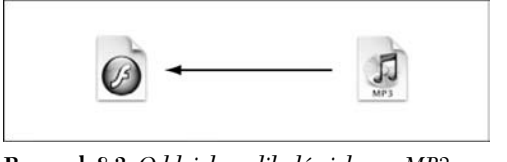
Każdy dźwięk odtwarzany z poziomu ActionScript wymaga utworzenia dla niego egzemplarza klasy `Sound`. Mając taki egzemplarz, możemy zastosować dla niego metodę `play()`, by odtworzyć dźwięk. W trakcie powstaje egzemplarz klasy `SoundChannel`, wyposażony we własności, dzięki którym możemy sterować dźwiękiem. Jedną z tych własności jest obiekt `SoundTransform`, który pozwala kontrolować globalne natężenie dźwięku oraz jego balans, czyli rozkład głośności na prawy i lewy głośnik.



Rysunek 8.1. *Umieszczenie dźwięku w klatce kluczowej na listwie czasowej to najprostszy sposób dołączenia dźwięku do filmu, niewymagający użycia ActionScript*



Rysunek 8.2. *Dźwięki zaimportowane do biblioteki mogą być odtwarzane w trakcie odtwarzania filmu także bez umieszczania ich na listwie czasowej, jeśli użyjemy ActionScript*



Rysunek 8.3. *Oddzielny plik dźwiękowy MP3 może zostać załadowany do filmu Flasha (SWF) i odtworzony za pomocą ActionScript*

Odtwarzanie dźwięków z biblioteki

Możesz zaimportować wszystkie potrzebne dźwięki do biblioteki podczas pracy w środowisku edycyjnym Flasha, a potem odtwarzać je w odpowiednich momentach w trakcie odgrywania filmu. Wymaga to jednak udostępnienia symboli dźwiękowych z biblioteki środowisku ActionScript. Robimy to tak samo, jak w przypadku symboli graficznych. W rozdziale 7. procedura ta została opisana dla symboli klipów filmowych oraz symboli bitmap. Polega to na utworzeniu nowej klasy dla symbolu, będącej rozszerzeniem jednej z już istniejących klas, dzięki czemu możemy tworzyć nowe egzemplarze danego symbolu dynamicznie, za pomocą funkcji konstruktora. W przypadku symboli dźwiękowych będziemy korzystać z rozszerzeń klasy `Sound`. Nazwę tworzonej klasy można ustalić w oknie *Właściwości połączenia*, po wybraniu polecenia *Łączenie* z menu opcji panelu *Biblioteka*.

Rysunek 8.4. *Wybierz polecenie Łączenie z menu opcji panelu Biblioteka dla każdego symbolu dźwiękowego, który chcesz kontrolować z poziomu ActionScript*

Aby przygotować symbol dźwiękowy z biblioteki do odtwarzania z ActionScript:

1. Zaimportuj do Flasha plik dźwiękowy, wybierając polecenie *Plik/Importuj/Importuj do biblioteki* i otwierając plik dźwiękowy.

   Wybrany plik dźwiękowy pojawi się w bibliotece filmu. Masz do wyboru następujące formaty pliku dźwiękowego: WAV (dla Windows) AIF (Mac OS) oraz MP3 (Windows i Mac OS). Więcej formatów może być dostępnych, gdy masz zainstalowany program QuickTime na swoim komputerze.

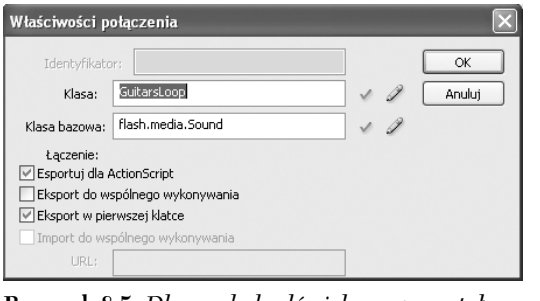
2. Zaznacz symbol dźwiękowy w panelu *Biblioteka*.

3. Z menu opcji panelu *Biblioteka* wybierz polecenie *Łączenie* (rysunek 8.4).

   Pojawia się okno dialogowe *Właściwości połączenia*.

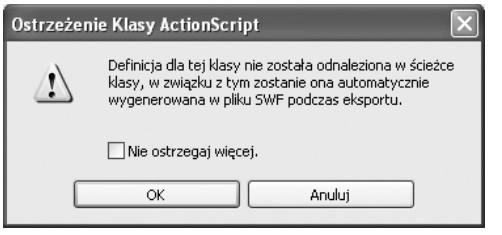
4. W polu *Łączenie* zaznacz opcję *Eksportuj dla ActionScript*. Pozostaw włączoną opcję *Eksport w pierwszej klatce*.

5. W polu tekstowym *Klasa* wpisz nazwę klasy. Dzięki tej nazwie będziesz mógł zidentyfikować dany dźwięk w ActionScript. W polu *Klasa bazowa* pozostaw klasę `flash.media.Sound`. Kliknij *OK* (rysunek 8.5).



Rysunek 8.5. *Dla symbolu dźwiękowego została utworzona nowa klasa GuitarLoop, dziedzicząca metody i własności klasy Sound*

Może pojawić się okno dialogowe z ostrzeżeniem, że klasa o takiej nazwie nie została znaleziona, będzie więc musiała zostać utworzona (rysunek 8.6). Kliknij *OK*. W naszym przykładzie nazwa nowej klasy to `GuitarLoop`. Jest ona częścią istniejącej klasy `Sound`, a więc dziedziczy wszystkie własności i metody klasy `Sound`. Nazwa nowej klasy będzie wykorzystywana przy tworzeniu nowych egzemplarzy tego konkretnego dźwięku w ActionScript. Upewnij się, że nazwa klasy nie zawiera kropek.



Rysunek 8.6. *Kliknij OK, by zamknąć to okienko ze zbędnym ostrzeżeniem. Informuje Cię ono, że zostanie utworzona nowa klasa*

Aby odtworzyć dźwięk z biblioteki:

1. Kontynuuj pracę nad plikiem z poprzedniego ćwiczenia. Zaznacz pierwszą klatkę na głównej listwie czasowej i otwórz panel *Operacje*.
2. W pierwszej linii utwórz nowy egzemplarz swojego symbolu dźwiękowego, posługując się nazwą klasy, którą dla niego utworzyłeś:

   ```
   var mySound:GuitarLoop = new
   ➥GuitarLoop();
   ```

   Zostaje utworzony nowy egzemplarz obiektu `Sound`, zawierający wybrany dźwięk z biblioteki.
3. Wpisz nazwę nowego egzemplarza `Sound`, kropkę, a potem metodę `play()` (rysunek 8.7).

   Dźwięk zaczyna być odtwarzany. Po jednokrotnym odtworzeniu dźwięku odtwarzanie zostanie wstrzymane.

```
var mySound:GuitarsLoop = new GuitarsLoop();
mySound.play();
```

Rysunek 8.7. *Zostaje utworzony nowy egzemplarz klasy GuitarLoop, o nazwie mySound („mój dźwięk"). Egzemplarz ten mieści w sobie dźwięk z biblioteki filmu. Następnie metoda play() zostaje użyta do odtworzenia tego dźwięku*

Wskazówka

- Metoda `play()` odtwarza egzemplarz dźwięku przy każdym wywołaniu, nawet gdy ten sam dźwięk już jest odtwarzany. To oznacza, że możemy mieć wiele nakładających się dźwięków w tym samym momencie filmu. Aby zapobiec tej sytuacji, użyj metody `stopAll()` klasy `SoundMixer`, zanim zarządzisz powtórne odtworzenie dźwięku. Ta technika gwarantuje, że dźwięk ucichnie, zanim zacznie być odtwarzany od początku.

Wczytywanie i odtwarzanie zewnętrznych dźwięków

Za każdym razem, gdy importujesz dźwięk do biblioteki filmu, zwiększa się rozmiar wynikowego pliku SWF. Dźwięki mogą zajmować bardzo dużo miejsca, nawet w przypadku stosowania kompresji MP3, więc warto zastanowić się nad przechowywaniem ich osobno. Najprościej można to zrobić, umieszczając każdy dźwięk w zewnętrznym pliku. Metoda `load()` wczytuje do filmu pliki MP3 (pozostałe formaty nie są dostępne). Takie podejście umożliwia podmianę pliku dźwiękowego bez potrzeby edycji dokumentu Flasha. Wyobraź sobie, że podkład muzyczny filmu zapisałeś w pliku *audio1.mp3*. Aby zmienić muzykę na inną, wystarczy zastąpić plik *audio1.mp3* innym plikiem, nadając mu tę samą nazwę; Flash zacznie odtwarzać nowy dźwięk.

Metoda `load()` wymaga podania tylko jednego parametru. Jest nim obiekt `URLRequest`, który zawiera ścieżkę dostępu do pliku MP3.

Aby załadować i odtworzyć zewnętrzny dźwięk:

1. Zadeklaruj i zainicjalizuj obiekt `URLRequest` za pomocą funkcji konstruktora `new URLRequest()`. Jako parametr wpisz ścieżkę do pliku MP3, którego chcesz użyć (rysunek 8.8).

```
var myRequest:URLRequest=new URLRequest ("music.mp3")
```

Rysunek 8.8. *Obiekt URLRequest definiuje ścieżkę adresową pliku, który chcesz załadować. W tym przypadku jest to plik dźwiękowy music.mp3, umieszczony w tym samym katalogu, w którym znajduje się film Flasha*

Ścieżka dostępu jest podawana w formie ciągu znaków, musisz więc ująć ją w cudzysłów. Możesz załadować plik MP3 z własnego dysku lub z internetu, używając adresu bezwzględnego. Jeśli plik MP3 znajduje się w tym samym katalogu, w którym jest film Flasha, wystarczy podać jego nazwę.

2. Zadeklaruj i zainicjalizuj obiekt `Sound` za pomocą funkcji konstruktora `new Sound()` (rysunek 8.9).

3. W kolejnej linii wpisz nazwę obiektu `Sound`, kropkę, a potem metodę `load()`. Użyj obiektu `URLRequest` jako parametru tej metody.

4. W kolejnej linii wpisz nazwę swojego obiektu `Sound`, kropkę i metodę `play()`. Możesz użyć dodatkowych parametrów (nie jest to obowiązkowe), by określić początek odtwarzania lub liczbę powtórzeń (rysunek 8.10).

   Gdy tylko film Flasha zostanie uruchomiony, wczyta plik MP3 i odtworzy go.

5. Zapisz plik Flasha w tym samym katalogu, w którym znajduje się plik MP3. Przetestuj film.

   Zaraz po rozpoczęciu odtwarzania filmu Flash odszukuje zewnętrzny plik MP3 za pomocą obiektu `URLRequest`, ładuje go i odtwarza równolegle z filmem.

```
var myRequest:URLRequest=new URLRequest ("music.mp3")
var mySound:Sound = new Sound()
```

Rysunek 8.9. *W drugiej linii kodu tworzymy obiekt klasy Sound o nazwie mySound*

```
var myRequest:URLRequest=new URLRequest ("music.mp3")
var mySound:Sound = new Sound()
mySound.load(myRequest)
mySound.play()
```

Rysunek 8.10. *Metoda load() pobiera plik dźwiękowy z miejsca wskazanego przez obiekt URLRequest, a metoda play() go odtwarza*

Sterowanie odtwarzaniem dźwięków

Metoda `play()` obiektu `Sound` przyjmuje trzy opcjonalne parametry. Pierwszy z nich to parametr przesunięcia w czasie. Jest to liczba oznaczająca, ile milisekund nagrania, licząc od początku trwania dźwięku, zostanie pominiętych. Odtwarzanie dźwięku można rozpocząć od samego początku lub od dowolnego późniejszego punktu. Gdy dźwięk trwa 20 sekund, nic nie stoi na przeszkodzie, by rozpocząć odtwarzanie od jego środka, czyli od 10. sekundy; wystarczy użyć metody `play(10000)`. Nie jest to opóźnienie rozpoczęcia odtwarzania o 10 sekund, ale natychmiastowe odtworzenie od 10. sekundy.

Drugi parametr określa liczbę odtworzeń dźwięku. Przekazanie wartości 2 spowoduje odtworzenie dźwięku dwukrotnie bez żadnych przerw między kolejnymi uruchomieniami. Wielokrotne powtarzanie (zapętlenie) jest zwykle stosowane dla dźwięków tak przygotowanych, by ich koniec pasował do początku, tak że moment przeskoku pomiędzy końcem a początkiem nagrania jest niezauważalny.

Trzecim parametrem metody `play()` jest obiekt `SoundTransform`. Zapewnia on kontrolę nad ogólną głośnością dźwięku oraz nad balansem głośności pomiędzy lewym a prawym głośnikiem. Obiekt `SoundTransform` zostanie dokładniej opisany w dalszej części rozdziału.

Jeśli w momencie wywoływania metody `play()` nie przekaże się żadnych parametrów, Flash odtworzy dźwięk jednokrotnie od samego początku.

Aby odtworzyć dźwięk od wybranego punktu:

- Użyj dla metody `play()` obiektu `Sound` pierwszego z jej parametrów, podając wartość wyrażoną w milisekundach.

  Dźwięk będzie odtwarzany od tego punktu w czasie (licząc w milisekundach; rysunek 8.11).

Aby ustalić liczbę powtórzeń:

- Użyj dla metody `play()` obiektu `Sound` drugiego z jej parametrów, podając liczbę powtórzeń.

  Dźwięk będzie odtwarzany tyle razy, ile sobie zażyczyłeś (rysunek 8.12).

Wskazówki

- Niestety, nie istnieje sposób, by poinformować metodę `play()` o chęci nieskończonego zapętlenia dźwięku. W takiej sytuacji jako wartość parametru zapętlenia możesz przekazać bardzo dużą wartość, na przykład 99999.

```
var myRequest:URLRequest=new URLRequest ("music.mp3")
var mySound:Sound = new Sound()
mySound.load(myRequest)
mySound.play(14000)
```

Rysunek 8.11. *W tym przykładzie dla metody play() użyto parametru o wartości 14000, co oznacza odtwarzanie dźwięku od 14. sekundy*

```
var myRequest:URLRequest=new URLRequest ("music.mp3")
var mySound:Sound = new Sound()
mySound.load(myRequest)
mySound.play(0,3)
```

Rysunek 8.12. *W tym przykładzie dla metody play() pierwszy parametr ustawiono na wartość 0, a drugi na 3, co oznacza trzykrotne odtworzenie dźwięku od samego początku*

Wstrzymywanie odtwarzania dźwięku

Odtwarzanie dźwięku można wstrzymać, stosując metodę klasy `SoundChannel`. Gdy wywołujemy metodę `play()` klasy `Sound`, tworzony jest egzemplarz obiektu `SoundChannel`. Dla każdego odtwarzanego dźwięku jest jeden egzemplarz `SoundChannel`.

Aby przypisać egzemplarz `SoundChannel` do zmiennej i móc odnosić się później do niego w ActionScript, użyj następującej instrukcji:

```
myChannel:SoundChannel=
➥mySound.play();
```

W tym przykładzie odtwarzany jest dźwięk dołączony do obiektu zwanego `mySound`. Zwrócony obiekt `SoundChannel` jest umieszczany w zmiennej o nazwie `myChannel`. Możesz teraz wstrzymać odtwarzanie dźwięku, stosując metodę `stop()` obiektu klasy `SoundChannel`:

```
myChannel.stop();
```

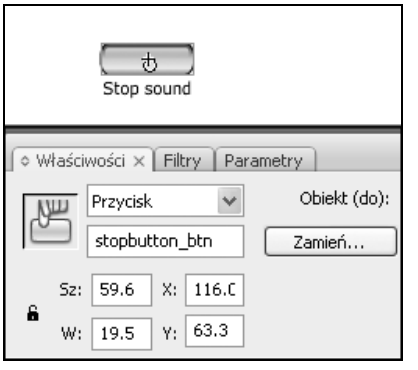
Aby zatrzymać odtwarzanie dźwięku:

1. Kontynuuj pracę nad plikiem z ćwiczenia „Aby załadować i odtworzyć zewnętrzny dźwięk".
2. Utwórz nowy symbol przycisku, umieść jego egzemplarz na scenie i nadaj mu nazwę w panelu *Właściwości* (rysunek 8.13).

   W naszym przykładzie użyjemy funkcji obsługującej zdarzenie kliknięcia tego przycisku, by wstrzymać odtwarzanie dźwięku.
3. Zaznacz pierwszą klatkę głównej listwy czasowej i otwórz panel *Operacje*.
4. Zastąp polecenie zawierające metodę `play()` następującą instrukcją:

   ```
   var myChannel:SoundChannel
   ➥=my.Sound.play()
   ```

   To polecenie odtwarza dźwięk i umieszcza obiekt `SoundChannel`, zwrócony przez metodę `play()`, w nowej zmiennej (typu `SoundChannel`) o nazwie `myChannel` (rysunek 8.14).



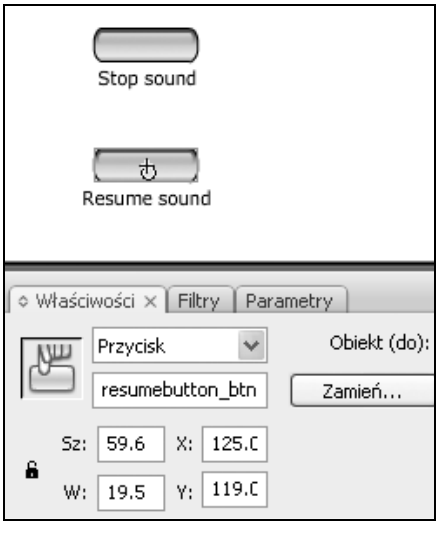
Rysunek 8.13. *Ten egzemplarz przycisku na scenie otrzymał nazwę stopbutton_btn*

```
var myRequest:URLRequest=new URLRequest ("music.mp3");
var mySound:Sound = new Sound();
mySound.load(myRequest);
var myChannel:SoundChannel=mySound.play();
```

Rysunek 8.14. *Gdy wywoływana jest metoda play() obiektu Sound, zwraca ona obiekt SoundChannel. Obiekt ten jest przypisywany do zmiennej myChannel*

```
stopbutton_btn.addEventListener(MouseEvent.CLICK, stopsound);
function stopsound(myevent:MouseEvent) {
    myChannel.stop();
}
```

Rysunek 8.15. *Funkcja obsługująca zdarzenie kliknięcia przycisku stopbutton_btn na scenie. Metoda stop(), która zatrzymuje odtwarzanie dźwięku, jest wywoływana dla obiektu SoundChannel, zwanego myChannel*



Rysunek 8.16. *Drugi egzemplarz przycisku na scenie otrzymuje nazwę resumebutton_btn*

5. Utwórz funkcję typu *event handler*, wykrywającą kliknięcie przycisku na scenie.
6. Wewnątrz nawiasów klamrowych funkcji wpisz nazwę obiektu `SoundChannel`, kropkę i metodę `stop()`, tak jak w poniższym przykładzie (rysunek 8.15).

   ```
   myChannel.stop();
   ```

7. Przetestuj film.

   Zewnętrzny dźwięk zaczyna być odtwarzany po uruchomieniu filmu. Kliknięcie przycisku spowoduje zakończenie odtwarzania.

Wskazówka

- Możesz również użyć metody `stopAll()` klasy `SoundMixer`. To spowoduje zatrzymanie odtwarzania wszystkich dźwięków. Użyj na przykład polecenia `SoundMixer.stopAll()`.

Wznawianie dźwięków

Możesz dowiedzieć się, jaka część dźwięku została już odtworzona, korzystając z własności `position` obiektu `SoundChannel`. Własność `position` zwraca bieżący czas odtwarzanego nagrania w milisekundach. Jest to bardzo przydatne, gdy chcesz zapamiętać moment, w którym odtwarzanie dźwięku zostało przerwane, a potem wznowić je od tego samego punktu.

Gdy użytkownik zatrzymuje odtwarzanie dźwięku, możesz przechwycić wartość własności `position` obiektu `SoundChannel` w tym momencie, przypisując tę wartość do zmiennej. Potem, wznawiając odtwarzanie dźwięku, przypiszesz przechwyconą wartość do pierwszego parametru metody `play()`obiektu `Sound`.

Aby wznowić odtwarzanie:

1. Kontynuując pracę nad plikiem z poprzedniego ćwiczenia, umieść na scenie kolejny egzemplarz przycisku i nadaj mu nazwę w panelu *Właściwości* (rysunek 8.16).

   W tym ćwiczeniu utworzymy funkcję obsługującą zdarzenie kliknięcia tego drugiego przycisku, aby umożliwić wznowienie odtwarzania dźwięku od punktu, w którym zostało przerwane.

2. Zaznacz pierwszą klatkę głównej listwy czasowej i otwórz panel *Operacje*.

3. Zacznij tworzenie dalszej części skryptu od zadeklarowania zmiennej do przechowywania wartości całkowitych (*integer*; rysunek 8.17).

   Zmienna ta będzie przechowywać bieżącą pozycję odtwarzania.

4. Wróć do funkcji *event handler*, utworzonej dla pierwszego przycisku, przeznaczonego do zatrzymania odtwarzania. Tuż przed metodą `stop()` wpisz dodatkowe polecenie, takie jak to poniżej (rysunek 8.18):

   ```
   pauseposition=myChannel.position;
   ```

   Tuż przed zatrzymaniem dźwięku bieżąca pozycja odtwarzania jest przypisywana do zmiennej.

5. Utwórz kolejną funkcję typu *event handler*, reagującą na kliknięcie drugiego przycisku umieszczonego na scenie.

   Zadaniem tej funkcji będzie wznowienie odtwarzania dźwięku.

6. Pomiędzy nawiasami klamrowymi funkcji wpisz następujące polecenie (rysunek 8.19):

   ```
   myChannel:SoundChannel=
   ↪mySound.play(pauseposition);
   ```

   Pozycja odtwarzania zatrzymanego dźwięku zostanie użyta jako pierwszy parametr metody `play()`, który decyduje o wartości przesunięcia. Dźwięk jest odtwarzany od miejsca, w którym się zatrzymał.

7. Przetestuj film.

   Zewnętrzny plik dźwiękowy zaczyna się odtwarzać. Kliknięcie pierwszego przycisku wstrzymuje odtwarzanie. Kliknięcie drugiego — wznawia je.

```
var myRequest:URLRequest=new URLRequest ("music.mp3");
var mySound:Sound = new Sound();
mySound.load(myRequest);
var myChannel:SoundChannel=mySound.play();
var pauseposition:int;
```

Rysunek 8.17. *Podświetlone polecenie deklaruje nową zmienną o nazwie pauseposition, która będzie przechowywać liczbę całkowitą*

```
stopbutton_btn.addEventListener(MouseEvent.CLICK, stopsound);
function stopsound(myevent:MouseEvent) {
    pauseposition = myChannel.position;
    myChannel.stop();
}
```

Rysunek 8.18. *Zanim odtwarzanie dźwięku zostanie przerwane, obecna pozycja odtwarzania (czas, jaki minął od początku nagrania, w milisekundach) jest przechwytywana i zapisywana w zmiennej pauseposition*

```
resumebutton_btn.addEventListener(MouseEvent.CLICK, resumesound);
function resumesound(myevent:MouseEvent) {
    myChannel=mySound.play(pauseposition);
}
```

Rysunek 8.19. *Funkcja obsługująca zdarzenie kliknięcia przycisku resumebutton_btn, umieszczonego na scenie. Metoda play() wykorzystuje zmienną pauseposition jako parametr, tak by odtwarzanie było wznawiane od momentu, w którym je przerwano*

Modyfikacja natężenia i balansu dźwięku

Flash oferuje pełną kontrolę nad głośnością i **balansem dźwięku** (siłą dźwięku podawaną do lewego i prawego głośnika). Nic więc nie stoi na przeszkodzie, by dać użytkownikowi możliwość sterowania głośnością lub też stosować zmiany głośności do uzyskania efektów specjalnych. Możesz na przykład zwiększać, a potem stopniowo zmniejszać natężenie ryku silnika, gdy w grze w wyścigi samochodowe wymija gracza inny zawodnik. Sterowanie balansem można zastosować w grze Pong, by odgłos uderzenia piłki w rakietkę dobiegał z głośnika po właściwej stronie.

Aby zmienić ogólną głośność oraz balans danego dźwięku, musisz wprowadzić trzeci parametr do metody `play()` (jak zapewne pamiętasz, pierwszy parametr decyduje o przesunięciu odtwarzania, a drugi — o liczbie powtórzeń). Trzeci parametr wymaga użycia obiektu klasy `SoundTransform`. Najpierw trzeba go więc utworzyć:

```
var newVolume:SoundTransform=new
↪SoundTransform()
```

Następnie trzeba ustawić odpowiednią własność tego obiektu — na przykład `volume` — na właściwą wartość, a potem użyć obiektu `SoundTransform` jako trzeciego parametru metody `play()`, tak jak w poniższym przykładzie:

```
newVolume.volume=.5;
mySound.play(0,0,newVolume);
```

Aby zmienić głośność lub balans dźwięku, który już jest odtwarzany, należy przypisać obiekt `SoundTransform` do własności `soundTransform` obiektu `SoundChannel`. Na przykład tak:

```
var newVolume:SoundTransform=new
↪SoundTransform();
newVolume.volume=.5;
myChannel.soundTransform=newVolume;
```

Pierwsze z tych poleceń tworzy nowy obiekt klasy `SoundTransform` o nazwie `newVolume` („nowa głośność"). Następnie własność `volume` tego obiektu zostaje odpowiednio zmieniona. Na koniec obiekt `SoundTransform` zostaje przypisany do własności `soundTransform` obiektu `SoundChannel`, związanego z dźwiękiem, który jest aktualnie odtwarzany.

Klasa `SoundTransform` jest wyposażona we własności, takie jak `volume` (głośność), `pan` (balans) oraz jeszcze kilka, służących do precyzyjnego manipulowania prawym i lewym kanałem. Ich opis znajdziesz w tabeli 8.1.

Tabela 8.1. *Właściwości obiektu SoundTransform*

Parametr	Wartość
`volume`	Wartość liczbowa; 0 oznacza zupełne wyciszenie, 1 to pełna głośność
`pan`	Wartość liczbowa oznaczająca balans głośników; −1 to maksymalne przesunięcie w lewo, 1 to maksymalne przesunięcie w prawo
`leftToLeft`	Wartość liczbowa (od 0 do 1) określająca, ile lewego kanału zostanie odtworzone w lewym głośniku
`leftToRight`	Wartość liczbowa (od 0 do 1) określająca, ile lewego kanału zostanie odtworzone w prawym głośniku
`rightToLeft`	Wartość liczbowa (od 0 do 1) określająca, ile prawego kanału zostanie odtworzone w lewym głośniku
`rightToRight`	Wartość liczbowa (od 0 do 1) określająca, ile prawego kanału zostanie odtworzone w prawym głośniku

Aby ustawić głośność dźwięku przed jego odtworzeniem:

1. Zadeklaruj i zainicjalizuj obiekt `URLRequest`, korzystając z konstruktora new `URLRequest()`, i wpisz w roli parametru ścieżkę do pliku MP3.
2. W następnej linii zadeklaruj i zainicjalizuj nowy obiekt `Sound`, korzystając z konstruktora `new Sound()`.
3. W kolejnym wierszu wpisz nazwę swojego obiektu `Sound`, kropkę i metodę `load()`. W roli parametru użyj obiektu `URLRequest`.

   Flash ładuje zewnętrzny plik MP3, wskazany przez obiekt `URLRequest`.
4. W kolejnej linii zadeklaruj i zainicjalizuj nowy obiekt `SoundTransform`, korzystając z konstruktora `new SoundTransform()`.

   Obiekt ten jest wyposażony we własności, których możemy użyć do sterowania głośnością i balansem dźwięku.
5. W kolejnej linii wpisz nazwę swojego obiektu `SoundTransform`, kropkę i własność, której chcesz użyć: `volume` dla głośności, `pan` dla balansu. Potem wstaw znak równości i wpisz odpowiednią wartość (rysunek 8.20).
6. W następnej linii wpisz nazwę obiektu `Sound`, kropkę i metodę `play()`. Jako parametry wpisz 0, 5 oraz nazwę obiektu `SoundTransform` (rysunek 8.21). Przypisz wartość zwróconą przez metodę `play()` do obiektu klasy `SoundChannel`.
7. Przetestuj film.

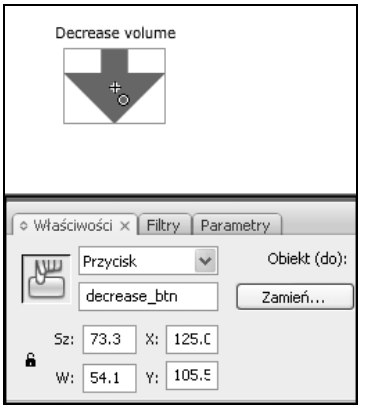
   Metoda `play()` odtwarza dźwięk, korzystając z obiektu `SoundTransform` do ustalenia poziomu głośności i (lub) balansu.

```
var myRequest:URLRequest=new URLRequest ("music.mp3");
var mySound:Sound = new Sound();
mySound.load(myRequest);

var newSetting:SoundTransform = new SoundTransform();
newSetting.volume=0.5;
```

Rysunek 8.20. *Własność głośności (volume) tego obiektu SoundTransform została ustawiona na połowę pełnej głośności*

```
var myRequest:URLRequest=new URLRequest ("music.mp3");
var mySound:Sound = new Sound();
mySound.load(myRequest);

var newSetting:SoundTransform = new SoundTransform();
newSetting.volume=0.5;

var myChannel:SoundChannel=mySound.play(0,5,newSetting);
```

Rysunek 8.21. *Obiekt SoundTransform o nazwie newSetting zostaje użyty jako trzeci parametr metody play(). Ten dźwięk będzie odtwarzany od początku, pięć razy, z głośnością zmniejszoną do połowy*



Rysunek 8.22. *Egzemplarz przycisku na scenie otrzymał nazwę decrease_btn*

Aby ustawić głośność dźwięku w trakcie odtwarzania:

1. Utwórz przycisk, umieść jego egzemplarz na scenie i nadaj mu nazwę w panelu *Właściwości* (rysunek 8.22).

   Utworzymy funkcję typu *event handler*, obsługującą kliknięcie tego przycisku. Przycisk umożliwi zmianę głośności dźwięku w trakcie odtwarzania.

2. Zaznacz pierwszą klatkę głównej listwy czasowej i otwórz panel *Operacje*. Zacznij od zadeklarowania i zainicjalizowania obiektu `URLRequest` za pomocą konstruktora `new URLRequest()`. Jako parametr podaj ścieżkę do zewnętrznego pliku MP3.

3. W kolejnej linii zadeklaruj i zainicjalizuj obiekt `Sound` za pomocą konstruktora `new Sound()`.

4. W następnym wierszu wpisz nazwę obiektu `Sound`, kropkę i metodę `load()`. Wpisz w nawiasach nazwę obiektu `URLRequest` jako parametr tej metody.

   Flash załaduje zewnętrzny plik MP3, którego adres jest zawarty w obiekcie `URLRequest`.

5. W kolejnej linii zadeklaruj i zainicjalizuj obiekt `SoundTransform` za pomocą konstruktora `new SoundTransform()`.

   Obiekt ten wyposażony jest we własności, których możemy użyć do sterowania głośnością i balansem dźwięku. W tym przykładzie egzemplarz obiektu `SoundTransform` otrzymał nazwę `newSetting` („nowe ustawienia”).

6. W kolejnej linii wpisz następujące polecenie:

```
var myChannel:SoundChannel
↪=mySound.play();
```

To polecenie odtwarza dźwięk i umieszcza zwrócony przez metodę `play()` obiekt klasy `SoundChannel` w nowej zmiennej, tu o nazwie `myChannel` (rysunek 8.23).

7. W następnej linii utwórz funkcję obsługującą zdarzenie (*event handler*), która będzie wykrywać kliknięcie przycisku na scenie. W nawiasach klamrowych funkcji umieść polecenie:

```
newSetting.volume-=0.1;
```

To oznacza, że przy każdym kliknięciu przycisku wartość przypisana własności `volume` obiektu `SoundTransform` zmniejszy się o 0,1.

8. W kolejnej linii, lecz wciąż w ramach funkcji *event handler*, dodaj polecenie:

```
myChannel.soundTransform=newSetting;
```

To polecenie przypisuje obiekt `SoundTransform` o nazwie `newSetting` do własności `soundTransform` odtwarzanego dźwięku, tak by zmniejszenie głośności doszło do skutku (rysunek 8.24).

9. Przetestuj film.

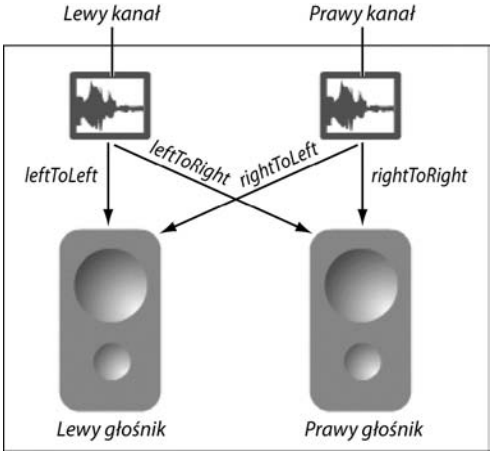
Zewnętrzny plik MP3 ładuje się i odtwarza. Przy każdym kliknięciu przycisku własność `volume` obiektu `SoundTransform` o nazwie `newSetting` zmniejsza się o 0,1. Następnie zmodyfikowany obiekt `newSetting` jest przypisywany do własności `soundTransform` odtwarzanego dźwięku, co powoduje obniżenie jego głośności.

```
var myRequest:URLRequest=new URLRequest ("music.mp3");
var mySound:Sound = new Sound();
mySound.load(myRequest);

var newSetting:SoundTransform = new SoundTransform();
var myChannel:SoundChannel=mySound.play();
```

Rysunek 8.23. *Odtwarzany jest zewnętrzny plik MP3 o nazwie music.mp3*

```
decrease_btn.addEventListener(MouseEvent.CLICK, decreaseVolume);
function decreaseVolume(myevent:MouseEvent):void {
    newSetting.volume-=0.1
    myChannel.soundTransform=newSetting;
}
```

Rysunek 8.24. *Funkcja obsługująca kliknięcie przycisku decrease_btn, umieszczonego na scenie. Każde kliknięcie powoduje obniżenie o 0,1 wartości przypisanej do własności volume obiektu klasy SoundTransform, zwanego newSetting. Obiekt SoundTransform jest przypisany do własności soundTransform obiektu SoundChannel, by można było zmieniać głośność w trakcie odtwarzania*



Rysunek 8.25. *Właściwości obiektu SoundTransform, określające rozkład dźwięku pochodzącego z lewego i prawego kanału w poszczególnych głośnikach. Właściwości te przyjmują wartości od 0 do 1*

Aby zamienić z sobą kanały dźwięku:

◆ Przypisz następujące wartości własnościom obiektu `SoundTransform`:

```
leftToLeft=0;
leftToRight=1;
rightToRight=0;
rightToLeft=1;
```

Gdy dokonasz takiej zmiany własności obiektu `SoundTransform` i przypiszesz ten obiekt do własności `soundTransform` obiektu `SoundChannel` lub użyjesz go w roli trzeciego parametru metody `play()`, kanały lewy i prawy zostaną zamienione: kanał prawy będzie odtwarzany z lewego głośnika, a lewy — z prawego (rysunek 8.25).

Wskazówka

■ Aby skrócić skrypt, możesz ustawić własności obiektu `SoundTransform` już w chwili jego zainicjalizowania. Na przykład polecenie `var myNewSettings:SoundTransform=new SoundTransform(.5,1)` stanowi odpowiednik następującej sekwencji poleceń:

```
var myNewSettings:SoundTransform=new
↪SoundTransform();
myNewSettings.volume=.5;
myNewSettings.pan=1;
```

Wykrywanie zdarzeń dźwięku

Można wykryć moment zakończenia odtwarzania dźwięku, używając zdarzenia `Event.SOUND_COMPLETE` klasy `SoundChannel`. Przyjrzyj się następującemu skryptowi:

```
myChannel.addEventListener
↪(Event.SOUND_COMPLETE, finished);
function finished(myevent:Event):void {
   //koniec odtwarzania
}
```

W tym przykładzie, gdy dźwięk związany z obiektem `SoundChannel` zostanie już w całości odtworzony, wywołana zostanie funkcja o nazwie `finished` („skończone") i wykonane zostaną wszystkie polecenia, jakie w tej funkcji umieścisz.

Zdarzenie `Event.SOUND_COMPLETE` umożliwia sterowanie dźwiękiem i jego integrację na wiele różnych sposobów. Wystarczy wyobrazić sobie szafę grającą, losowo wybierającą piosenki z banku albumów. Gdy zakończy się jedna piosenka, Flash wie, że nadszedł czas na rozpoczęcie kolejnej. Inny przykład to prezentacja biznesowa, która przechodzi do kolejnego slajdu dopiero po zakończeniu narracji. W poniższym ćwiczeniu wykorzystamy zdarzenie końca odtwarzania do ustalenia momentu, w którym Flash powinien załadować i odtworzyć kolejny plik.

Aby wykryć zakończenie odtwarzania dźwięku:

1. Zadeklaruj i zainicjalizuj obiekt `URLRequest` za pomocą funkcji konstruktora `new URLRequest()` i wpisz jako parametr adres zewnętrznego pliku MP3.
2. W kolejnej linii zadeklaruj i zainicjalizuj obiekt `Sound` za pomocą funkcji konstruktora `new Sound()`.
3. W kolejnej linii wpisz nazwę obiektu `Sound`, kropkę i metodę `load()`. Jako parametr tej metody podaj obiekt `URLRequest`.

4. W nowej linii wpisz nazwę obiektu `Sound`, kropkę i metodę `play()` bez parametrów. Przypisz zwróconą wartość do nowego obiektu `SoundChannel` (rysunek 8.26).
5. W kolejnej linii utwórz odbiornik zdarzeń w obiekcie `SoundChannel`, nastawiony na wykrycie zdarzenia `Event.SOUND_COMPLETE` (rysunek 8.27).
6. Utwórz funkcję uruchamianą przez zdarzenie `Event.SOUND_COMPLETE` (rysunek 8.28).

Gdy odtwarzanie dźwięku się zakończy, funkcja zostanie wywołana. W tym przypadku działanie funkcji polega na utworzeniu nowego obiektu `Sound`, który załaduje i odtworzy inny dźwięk.

Wskazówka

- Jeśli dźwięk jest zapętlony (powtarzany wielokrotnie), zdarzenie `Event.SOUND_COMPLETE` zostanie zgłoszone po ostatniej pętli.

```
var myRequest:URLRequest=new URLRequest ("music1.mp3");
var mySound:Sound = new Sound();
mySound.load(myRequest);
var myChannel:SoundChannel=mySound.play();
```

Rysunek 8.26. *Odtwarzany jest zewnętrzny plik MP3 o nazwie music1.mp3*

```
var myRequest:URLRequest=new URLRequest ("music1.mp3");
var mySound:Sound = new Sound();
mySound.load(myRequest);
var myChannel:SoundChannel=mySound.play();

myChannel.addEventListener(Event.SOUND_COMPLETE, finished);
```

Rysunek 8.27. *Odbiornik zdarzeń wykrywa zakończenie ładowania i wywołuje (tu jeszcze niezdefiniowaną) funkcję o nazwie finished*

```
var myRequest:URLRequest=new URLRequest ("music1.mp3");
var mySound:Sound = new Sound();
mySound.load(myRequest);
var myChannel:SoundChannel=mySound.play();

myChannel.addEventListener(Event.SOUND_COMPLETE, finished);

function finished(myevent:Event) {
    var myRequest2:URLRequest=new URLRequest ("music2.mp3");
    var mySound2:Sound = new Sound();
    mySound2.load(myRequest2);
    myChannel2=mySound2.play();
}
```

Rysunek 8.28. *Funkcja o nazwie finished rozpoczyna odtwarzanie drugiego pliku dźwiękowego*

Wykorzystanie znaczników informacyjnych plików MP3

Pliki MP3 to obecnie najpopularniejszy format cyfrowy przechowywania muzyki. Kompresja MP3 pozwala znacząco zmniejszyć rozmiar pliku przy jednoczesnym zachowaniu bardzo wysokiej jakości, niemal takiej jak przy odtwarzaniu płyt kompaktowych. Dodatkową zaletą plików MP3 jest możliwość „zaszycia" w ich wnętrzu dodatkowych informacji na temat muzyki — **metadanych**. Pierwsza wersja metadanych (tak zwanych znaczników ID3) była umieszczana na końcu pliku MP3. Zawierała takie informacje, jak autor, tytuł piosenki, nazwa albumu, rok wydania, komentarz i gatunek muzyczny. Informacje znajdujące się na końcu pliku były odczytywane przez większość odtwarzaczy MP3.

Po pewnym czasie powstała obowiązująca obecnie druga wersja ID3. Jedną z jej głównych cech jest przeniesienie informacji na początek pliku MP3, by usprawnić obsługę strumieniowania. Dodatkowo pojawiło się wiele nowych pól, między innymi informacja na temat kompozytora, dyrygenta, typ medium, informacja o prawach autorskich i data nagrania.

Flash potrafi odczytywać znaczniki ID3v2 plików MP3. Każdy element informacji (nazywany **znacznikiem**) odpowiada konkretnej właściwości obiektu `id3`, będącego częścią obiektu `Sound`. Przykładowo, `mySound_sound.id3.TALB` zawiera nazwę albumu. Tabela 8.2 wymienia wszystkie znaczniki obsługiwane przez Flasha za pośrednictwem własności obiektu `Sound`.

W jaki sposób pobiera się znaczniki ID3? Trzeba utworzyć funkcję obsługi zdarzenia, które jest wywoływane, gdy dostępne są znaczniki dla dźwięku odtwarzanego metodą `play()`. Służy do tego zdarzenie `Event.ID3`. Jest to jedyny sposób na odczytanie tych informacji.

W kolejnym ćwiczeniu załadujesz zewnętrzny plik MP3 i wyświetlisz zawarte w nim informacje w oknie *Wyjście*.

Tabela 8.2. *Znaczniki ID3v2*

Właściwość	Opis
`id3.COMM`	komentarz
`id3.TALB`	tytuł albumu, filmu lub nazwa przedstawienia
`id3.TBPM`	taktowanie
`id3.TCOM`	kompozytor
`id3.TCOP`	informacja o prawach autorskich
`id3.TDAT`	data
`id3.TDLY`	opóźnienie listy odtwarzania
`id3.TENC`	autor kodowania
`id3.TEXT`	autor słów
`id3.TFLT`	typ pliku
`id3.TIME`	czas
`id3.TIT1`	opis grupy zawartości
`id3.TIT2`	tytuł lub nazwa piosenki
`id3.TIT3`	podtytuł, dokładniejszy opis
`id3.TKEY`	klucz początkowy
`id3.TLAN`	języki
`id3.TLEN`	długość
`id3.TMED`	typ medium
`id3.TOAL`	oryginalny album, film lub nazwa przedstawienia
`id3.TOFN`	oryginalna nazwa pliku
`id3.TOLY`	pierwotny autor tekstu
`id3.TOPE`	pierwotny wykonawca
`id3.TORY`	pierwotna data wydania
`id3.TOWN`	właściciel pliku, licencja
`id3.TPE1`	główny wykonawca lub solista
`id3.TPE2`	zespół, orkiestra, akompaniament
`id3.TPE3`	dyrygent, reżyser dźwięku
`id3.TPE4`	zaaranżowane, zmiksowane lub w inny sposób zmienione przez
`id3.TPOS`	część zbioru
`id3.TPUB`	wydawca
`id3.TRCK`	numer ścieżki lub pozycja w zbiorze
`id3.TRDA`	data nagrania
`id3.TRSN`	nazwa radiowej stacji internetowej
`id3.TRSO`	właściciel radiowej stacji internetowej
`id3.TSIZ`	rozmiar
`id3.TSRC`	międzynarodowy, standaryzowany kod nagrania (ISRC)
`id3.TSSE`	oprogramowanie lub sprzęt użyty do kodowania
`id3.TYER`	rok
`id3.WXXX`	łącze URL

Aby pobrać informacje na temat pliku MP3:

1. Zadeklaruj i zainicjalizuj obiekt klasy `URLRequest`, używając konstruktora `new URLRequest()`. Jako parametr podaj ścieżkę do pliku MP3.
2. Zadeklaruj i zainicjalizuj obiekt klasy `Sound`, używając konstruktora `new Sound()`.
3. Wpisz nazwę utworzonego obiektu `Sound`, kropkę i metodę `load()`. W nawiasie jako parametr podaj wcześniej utworzony obiekt `URLRequest`.

   Flash ładuje zewnętrzny plik MP3, którego adres i nazwa zostały zdefiniowane w obiekcie `URLRequest`.
4. W nowym wierszu ponownie wpisz nazwę obiektu `Sound` oraz kropkę. Wpisz metodę `play()` bez parametrów (rysunek 8.29).

   Metoda `play()` odtwarza dźwięk.
5. Dodaj odbiornik zdarzeń do obiektu `Sound`. Nastaw go na wykrywanie zdarzenia `Event.ID3`, tak jak w poniższym przykładzie:

```
mySound.addEventListener(Event.ID3,
➥gotmedata);
```

   Gdy Flash otrzyma dane ID3 od załadowanego pliku MP3, zdarzenie to uruchomi funkcję o nazwie `gotmedata` („podaj dane").
6. Utwórz funkcję o nazwie `gotmedata`, wywoływaną przez zdarzenie `Event.ID3`. Pomiędzy nawiasami klamrowymi funkcji umieść polecenie `trace`, przeznaczone do wyświetlenia własności `id3`, tak jak w poniższym przykładzie (rysunek 8.30):

```
function gotmedata(myevent:
➥Event) : void{
trace("title="+mySound.id3.TIT2);
}
```

   W tym przykładzie polecenie `trace` wyświetla informację na temat tytułu piosenki zapisanej w pliku MP3. Do podanej informacji dołącza objaśnienie o brzmieniu *title=* („tytuł to...").

```
var myRequest:URLRequest=new URLRequest ("music.mp3");
var mySound:Sound = new Sound();
mySound.load(myRequest);
mySound.play();
```

Rysunek 8.29. *Odtwarzany jest zewnętrzny plik MP3 o nazwie music.mp3*

```
mySound.addEventListener(Event.ID3, gotmetadata);
function gotmetadata(myevent:Event):void {
    trace( "title="+mySound.id3.TIT2);
}
```

Rysunek 8.30. *Zdarzenie Event.ID3 zachodzi, gdy dostarczane są metadane z pliku MP3. Własność id3.TIT2 odnosi się do tytułu utworu*

7. Dodaj więcej poleceń `trace` pomiędzy nawiasami klamrowymi funkcji, tak by odczytana została cała informacja, jakiej potrzebujesz (rysunek 8.31).
8. Zapisz swój plik FLA w miejscu, z którego będzie potrafił znaleźć plik MP3, wskazany przez obiekt `URLRequest`[1].

   Testując film w środowisku edycyjnym Flasha, możesz odczytać informacje pochodzące ze znaczników ID3 w oknie *Wyjście* (rysunek 8.32).

Wskazówki

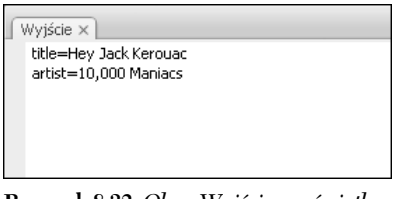
- Używając dynamicznych pól tekstowych, można wyświetlić użytkownikowi wszystkie potrzebne informacje, zawarte w znacznikach ID3, na scenie (a nie w oknie *Wyjście*). Więcej informacji na temat dynamicznych pól tekstowych znajduje się w rozdziale 10.
- Gdy plik MP3 zawiera znaczniki ID3v1 i ID3v2, zdarzenie `Event.ID3` zostanie zgłoszone dwa razy.
- Aby zobaczyć znaczniki ID3 plików MP3 poza Flashem, wykonaj następujące działania:

  W systemie Windows XP: kliknij prawym przyciskiem myszy plik MP3. Wybierz *Właściwości/Podsumowanie/Zaawansowane*.

  W systemie Mac OS X: w iTunes wybierz z listy nazwę piosenki i naciśnij klawisze *Command+I*.

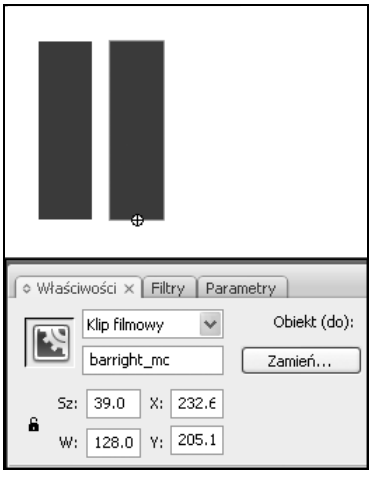
```
var myRequest:URLRequest=new URLRequest ("music.mp3");
var mySound:Sound = new Sound();
mySound.load(myRequest);
mySound.play();

mySound.addEventListener(Event.ID3, gotmetadata);
function gotmetadata(myevent:Event):void {
    trace( "title="+mySound.id3.TIT2);
    trace( "artist="+mySound.id3.artist);
}
```

Rysunek 8.31. *Dwie instrukcje trace umożliwiają wyświetlenie nazwiska wykonawcy i tytułu piosenki w oknie Wyjście, w trakcie testowania filmu, gdy nastąpi zdarzenie Event.ID3*



Rysunek 8.32. *Okno Wyjście, wyświetlane przy testowaniu filmu*

[1] Ważne tylko w przypadku stosowania względnych ścieżek adresowych; więcej informacji na ten temat znajduje się w rozdziale 5. — *przyp. tłum.*



Rysunek 8.33. *Dwa prostokątne klipy filmowe umieszczone na scenie: lewy o nazwie barleft_mc i prawy o nazwie barright_mc. Punkty zaczepienia tych obiektów są wyrównane do ich dolnej krawędzi*

Wizualizacja dźwięku

Prawdopodobnie widywałeś już graficzną reprezentację dźwięku w postaci fali lub wznoszących się w górę ostrych kolców, wyglądających jak postrzępione pasmo górskie, albo w postaci wibrujących linii i zmieniających się, tęczowych kolorów. Takie efekty można zobaczyć zarówno na wygaszaczach, jak i na pokazach laserowych typu „światło i dźwięk". Te efekty graficzne są zawsze związane z jakimś określonym parametrem dźwięku. Wraz ze zmianą dźwięku zmienia się również obraz, co pozwala uzyskać bezpośrednią, graficzną reprezentację tego, co słyszymy. Ten typ wizualizacji jest dobrym uzupełnieniem odtwarzanego dźwięku, a w razie wystąpienia problemów technicznych stanowi też użyteczną informację; dzięki niemu dowiadujemy się, czy w danym czasie jest odtwarzany jakiś dźwięk.

Wizualizację dźwięku, może nieco prostszą od opisanych, możesz z powodzeniem wykonać sam we Flashu. Klasa `SoundChannel` wyposażona jest w dwie własności, `leftPeak` oraz `rightPeak`, które przechowują informację na temat głośności dźwięku w lewym i prawym kanale w każdej chwili odtwarzania. Pobierając te dane dostatecznie często, na przykład za pomocą zdarzenia `Event.ENTER_FRAME` lub obiektu `Timer`, możesz wykonać na ich podstawie wizualizację. Możesz na przykład skalować pionowe paski odpowiednio do natężenia dźwięku w każdym kanale.

W kolejnym ćwiczeniu załadujemy i odtworzymy zewnętrzny plik MP3, a pobrane wartości własności `leftPeak` oraz `rightPeak` przypiszemy do własności `scaleY` dwóch prostokątnych klipów filmowych.

Aby wykonać wizualizację natężenia dźwięku w lewym i prawym kanale:

1. Utwórz klip filmowy z pionowym prostokątem, umieszczając punkt zaczepienia na dolnej krawędzi tego prostokąta. Umieść na scenie dwa egzemplarze tego klipu. Każdemu z nich nadaj indywidualną nazwę w panelu *Właściwości* (rysunek 8.33).

   Te dwa pionowe paski będą zmieniać swą wysokość tak, by odzwierciedlało to zmiany natężenia dźwięku w każdym z kanałów.

2. Jak w poprzednich ćwiczeniach, w panelu *Operacje* utwórz obiekt `URLRequest` za pomocą konstruktora `new URLRequest()`, podając jako parametr ścieżkę adresową pliku MP3.

3. Zadeklaruj i zainicjalizuj obiekt `Sound` za pomocą konstruktora `new Sound()`.

4. Wpisz nazwę obiektu `Sound`, kropkę i metodę `load()`. Podaj nazwę obiektu `URLRequest` jako jej parametr.

   Flash ładuje zewnętrzny plik MP3, zgodnie ze ścieżką podaną w obiekcie `URLRequest`.

5. Wywołaj metodę `play()` dla obiektu `Sound` i przypisz zwróconą wartość do nowej zmiennej typu `SoundChannel`:

```
var myChannel:SoundChannel=
↪mySound.play();
```

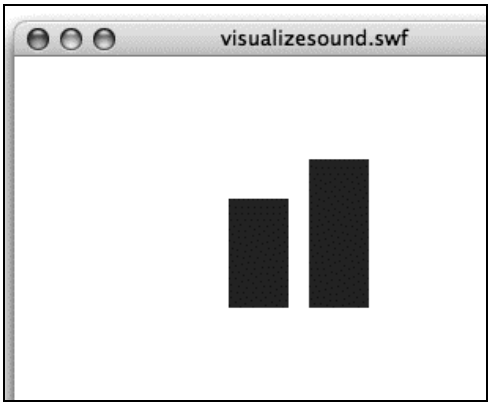
   Flash odtwarza dźwięk i tworzy powiązany z tym dźwiękiem obiekt `SoundChannel` (rysunek 8.34).

```
var mySound:Sound = new Sound();
var myRequest:URLRequest = new URLRequest("music.mp3");
mySound.load(myRequest);

var myChannel:SoundChannel= mySound.play();
```

Rysunek 8.34. *Odtwarzany jest zewnętrzny plik MP3, zwany music.mp3*

```
stage.addEventListener(Event.ENTER_FRAME, everyframe);
function everyframe(event:Event):void {
    barleft_mc.scaleY=(myChannel.leftPeak);
    barright_mc.scaleY=(myChannel.rightPeak);
}
```

Rysunek 8.35. *Zdarzenie Event.ENTER_FRAME wywołuje regularne skalowanie obu prostokątnych klipów, odpowiednio do bieżących wartości własności leftPeak i rightPeak obiektu Sound*



Rysunek 8.36. *Dwa prostokątne klipy zwiększają i zmniejszają swą wysokość w sposób zsynchronizowany z dźwiękiem*

6. Dodaj do sceny odbiornik zdarzeń (*event listener*), wykrywający zdarzenie `Event.ENTER_FRAME`:

```
stage.addEventLIstener
➥(Event.ENTER_FRAME, everyframe);
```

   Flash wywołuje funkcję `everyframe` („każda klatka") z częstotliwością wyznaczoną przez prędkość odtwarzania filmu.

7. Utwórz funkcję wywoływaną w odpowiedzi na zdarzenie `Event.ENTER_FRAME`.

8. W nawiasach klamrowych funkcji umieść polecenia, które zmienią wygląd klipów filmowych, tak jak w poniższym przykładzie (rysunek 8.35):

```
everyframe(event:Event):void{
    barleft_mc.scaleY=
    ➥(myChannel.leftPeak);
    barright_mc.scaleY=
    ➥(myChannel.rightPeak);
}
```

   Wartości liczbowe własności `leftPeak` i `rightPeak` zmieniają się w zakresie od 0 do 1. Po przypisaniu tych wartości do własności `scaleY` klipów filmowych tak samo zmienia się teraz współczynnik pionowego skalowania prostokątów.

9. Upewnij się, że zewnętrzny plik MP3 znajduje się w miejscu, w którym film Flasha może go odnaleźć, na podstawie informacji zawartej w obiekcie `URLRequest`. Przetestuj film (*Sterowanie/Testuj film*; rysunek 8.36).

Wskazówka

- Flash przewiduje także bardziej skomplikowany sposób obrazowania danych dźwiękowych; służy do tego klasa `computeSpectrum()` klasy `SoundMixer`. Metoda ta zwraca dane na temat częstotliwości fali, uzyskane z pomiaru wysokości tonów (niskie zakresy częstotliwości odpowiadają niskim tonom, a wysokie — wysokim). Więcej informacji na ten temat znajdziesz w pomocy Flasha. Patrz: *Help/Programming ActionScript 3.0/Working with Sound/Accessing raw sound data.*

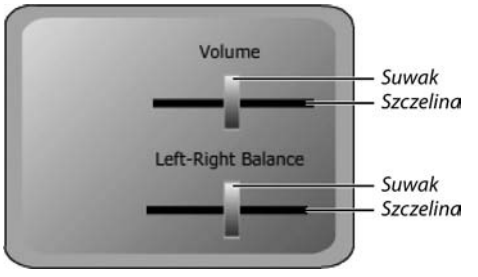
Tworzenie dynamicznego sterowania dźwiękiem

Jednym z najczęstszych zastosowań klasy `Sound` i klas z nią związanych jest umożliwienie użytkownikom sterowania głośnością i balansem odtwarzanych dźwięków. Najczęściej do zmiany głośności używa się klipu filmowego z możliwością przeciągania — powstaje suwak, którego położenie można powiązać z wartością własności `volume` własności `soundTransform` obiektu `SoundChannel`. Każda zmiana położenia suwaka pociąga za sobą wywołanie metody zmiany głośności.

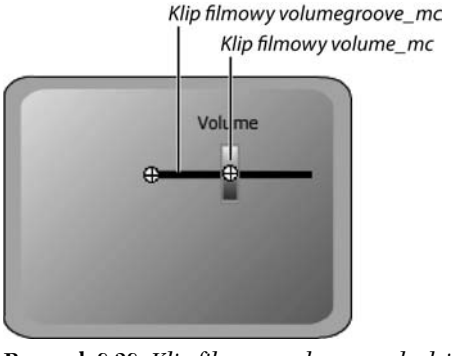
By utworzyć suwak zmiany głośności, trzeba utworzyć dwa elementy: symbol gałki suwaka i symbol prowadnicy dla suwaka (rysunek 8.37). Najpierw utwórz klip filmowy o nazwie `groove_mc` („prowadnica"). Aby ułatwić sobie zadanie, ustal długość szczeliny, w której przesuwa się gałka, na 100 pikseli. Punkt zaczepienia ustaw po lewej stronie prostokąta. Zastosowanie szerokości tego klipu równej 100 pikseli pozwoli łatwo skorelować położenie suwaka z wartością własności `volume`. Aby utworzyć gałkę suwaka z możliwością przeciągania, utwórz klip filmowy o nazwie `slider_mc` i wywołaj dla niego metodę `startDrag()` z ograniczeniem ruchu do obszaru klipu `groove_mc`.

Aby utworzyć interfejs do sterowania głośnością i balansem dźwięku:

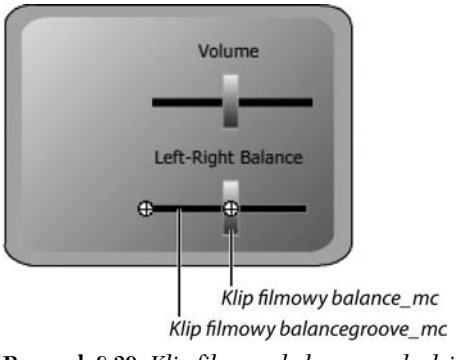
1. Utwórz klip filmowy z szerokim, lecz bardzo niskim prostokątem o szerokości 100 pikseli. Punkt zaczepienia umieść po lewej stronie prostokąta.
2. Umieść egzemplarz klipu na scenie i w panelu *Właściwości* nadaj mu nazwę `volumegroove_mc`.
3. Utwórz kolejny klip filmowy dla gałki.
4. Umieść klip gałki suwaka nad klipem `volumegroove_mc` i nadaj mu nazwę `volume_mc` w panelu *Właściwości* (rysunek 8.38).



Rysunek 8.37. *Składniki suwaków zapewniających zmianę głośności odtwarzania dźwięków oraz balansu. Na to urządzenie składają się dwie gałki i dwie szczeliny, wzdłuż których można je przeciągać*



Rysunek 8.38. *Klip filmowy volume_mc będzie można przeciągać wzdłuż klipu o nazwie volumegroove_mc. Punkt zaczepienia klipu volumegroove_mc musi znajdować się po lewej stronie*



Rysunek 8.39. *Klip filmowy balance_mc będzie można przeciągać wzdłuż klipu o nazwie balancegroove_mc. Punkt zaczepienia klipu balancegroove_mc musi znajdować się po lewej stronie*

5. Powtórz kroki 2. – 4. tego ćwiczenia, lecz tym razem nazwij szczelinę `balancegroove_mc`, a gałkę — `balance_mc`.

   Mamy teraz dwa suwaki, które będziemy przeciągać; na jednym będzie można ustawiać głośność, a na drugim — balans dźwięku (rysunek 8.39).

6. W panelu *Operacje* utwórz nowy obiekt `Rectangle`, którego lewy górny narożnik jest dopasowany do punktu zaczepienia klipu `volumegroove_mc`, szerokość pasuje do szerokości tego klipu, a wysokość jest ustawiona na 1, tak jak w tym przykładzie:

```
var myvolumebounds:Rectangle=new
↪Rectangle(volumegroove_mc.x,
↪volumegroove_mc.y,
↪volumegroove_mc.width, 1);
```

   Ten prostokąt posłuży do ograniczenia obszaru, w którym dozwolone jest przeciąganie gałki suwaka.

7. W następnej linii utwórz drugi prostokąt (`Rectangle`), pasujący do położenia i do szerokości klipu filmowego `balancegroove_mc` (rysunek 8.40).

8. W kolejnej linii utwórz odbiornik zdarzeń, wykrywający wciśnięcie myszy (`MouseEvent.↪MOUSE_DOWN`) nad klipem `volume_mc`.

   Zdarzenie `MOUSE_DOWN` wywoła akcję `startDrag` dla tej gałki suwaka.

9. W kolejnej linii utwórz odbiornik zdarzeń, wykrywający puszczenie myszy (`MouseEvent.↪MOUSE_UP`) nad klipem `volume_mc`.

   Zdarzenie `MOUSE_UP` wywoła akcję `stopDrag` dla tej gałki suwaka.

10. Powtórz kroki 8. – 9., tworząc odbiorniki zdarzeń `MOUSE_DOWN` i `MOUSE_UP` dla klipu `balance_mc`. Odbiorniki zdarzeń `MOUSE_UP` dla obu klipów, `volume_mc` i `balance_mc`, będą wywoływać tę samą funkcję (rysunek 8.41).

```
var myvolumebounds:Rectangle=new Rectangle(volumegroove_mc.x, volumegroove_mc.y, volumegroove_mc.width, 1);
var mybalancebounds:Rectangle=new Rectangle(balancegroove_mc.x, balancegroove_mc.y, balancegroove_mc.width, 1);
```

Rysunek 8.40. *Tworzone są dwa obiekty Rectangle, których zadaniem będzie ograniczyć możliwość przeciągania gałek suwaków. Każdy prostokąt jest wyrównany do lewej krawędzi szczeliny, ma szerokość równą szczelinie i tylko jeden piksel wysokości*

```
volume_mc.addEventListener(MouseEvent.MOUSE_DOWN, startdragvolume);
volume_mc.addEventListener(MouseEvent.MOUSE_UP, stopdrag);
balance_mc.addEventListener(MouseEvent.MOUSE_DOWN, startdragbalance);
balance_mc.addEventListener(MouseEvent.MOUSE_UP, stopdrag);
```

Rysunek 8.41. *Metody addEventListener() użyte dla zdarzeń MOUSE_DOWN i MOUSE_UP posłużą do rozpoczęcia i zakończenia akcji przeciągania*

11. W kolejnych liniach skryptu utwórz funkcje typu *event handler*, które będą rozpoczynać przeciąganie oraz je kończyć. Dla każdej z wywołanych metod `startDrag()` użyj odpowiedniego obiektu `Rectangle` jako parametru, ograniczając obszar przeciągania (rysunek 8.42).

Skrypt obsługujący przeciąganie suwaków jest już gotowy. W kolejnym ćwiczeniu będziemy kontynuować pracę nad tym plikiem i zajmiemy się wprowadzeniem dźwięku oraz połączeniem jego głośności i balansu z odpowiednimi suwakami.

Aby skorelować położenie gałek suwaków z głośnością i balansem dźwięku:

1. W panelu *Operacje*, w kolejnych liniach zainicjalizuj obiekt `URLRequest`, obiekt `Sound`, obiekt `SoundChannel` oraz obiekt `SoundTransform` (rysunek 8.43).
2. Wywołaj metodę `load()` obiektu `Sound`, używając obiektu `URLRequest` w roli parametru. Następnie wywołaj metodę `play()` obiektu `Sound` i przypisz zwróconą wartość do obiektu `SoundChannel` (rysunek 8.44).

```
function startdragvolume(myevent:MouseEvent):void {
    myevent.target.startDrag(false, myvolumebounds);
}
function startdragbalance(myevent:MouseEvent):void {
    myevent.target.startDrag(false, mybalancebounds);
}
function stopdrag(myevent:MouseEvent):void {
    myevent.target.stopDrag();
}
```

Rysunek 8.42. *Funkcje event handler dla akcji przeciągania i upuszczania obu gałek suwaków. Dla każdej z metod startDrag() użyto obiektu Rectangle jako parametru, ograniczając możliwość przeciągania do kierunku poziomego na szerokość szczeliny*

```
var myRequest:URLRequest=new URLRequest ("music.mp3");
var mySound:Sound = new Sound();
var myChannel:SoundChannel;
var newSetting:SoundTransform = new SoundTransform();
```

Rysunek 8.43. *Ta część skryptu tworzy obiekty potrzebne do załadowania dźwięku i kontrolowania go*

```
mySound.load(myRequest);
myChannel=mySound.play();
```

Rysunek 8.44. *Kolejna część skryptu odtwarza dźwięk*

3. W kolejnej linii utwórz odbiornik zdarzeń, wykrywający zdarzenie `Event.ENTER_FRAME`.

4. W kolejnej linii utwórz funkcję, uruchamianą zdarzeniem `Event.ENTER_FRAME`.

   Ta funkcja będzie wywoływana nieustannie, w każdej klatce filmu, wykorzystamy ją więc do utworzenia połączenia między gałkami suwaków a głośnością i balansem odtwarzanego dźwięku.

5. Pomiędzy nawiasami klamrowymi funkcji umieść następujące polecenie:

```
var newvolume:Number =
↪(volume_mc.x-volumegroove_mc.x)/100;
```

   Współrzędna pozioma położenia klipu szczeliny zostaje odjęta od współrzędnej poziomej aktualnego położenia gałki suwaka głośności (`volume_mc`), co oznacza, że uzyskamy liczbę z zakresu od 0 do 100. Dzieląc rezultat tego odejmowania przez 100, otrzymujemy wartość z zakresu 0 do 1, którą będziemy mogli przypisać własności `volume` odtwarzanego dźwięku. Uzyskana wartość jest przechowywana w zmiennej o nazwie `newvolume` („nowa głośność").

6. W następnej linii, nadal w nawiasach klamrowych funkcji, dodaj kolejne polecenie:

```
var newbalance:Number =
↪((balance_mc.x-
↪balancegroove_mc.x)/50)-1;
```

   Współrzędna pozioma położenia klipu szczeliny zostaje odjęta od współrzędnej poziomej aktualnego położenia gałki suwaka balansu (`balance_mc`), co oznacza, że uzyskamy liczbę z zakresu od 0 do 100. Dzieląc rezultat tego odejmowania przez 50, uzyskujemy zakres wartości od 0 do 2, a odejmując od tego 1, uzyskamy zakres od −1 do 1. Jest to odpowiedni zakres wartości dla własności `pan` dźwięku. Rezultat obliczeń jest przechowywany w zmiennej o nazwie `newbalance` („nowy balans").

7. Nadal w ramach funkcji przypisz nowe wartości własnościom `volume` i `pan` obiektu `SoundTransform` (w tym przykładzie jest to obiekt o nazwie `newSetting`, czyli „nowe ustawienia"). Na koniec przypisz obiekt `SoundTransform` do własności `soundTransform` bieżącego dźwięku (czyli obiektu `SoundChannel`, w tym przykładzie noszącego nazwę `myChannel`; rysunek 8.45).

```
newSetting.volume=newvolume;
newSetting.pan=newbalance;
myChannel.soundTransform=newSetting;
```

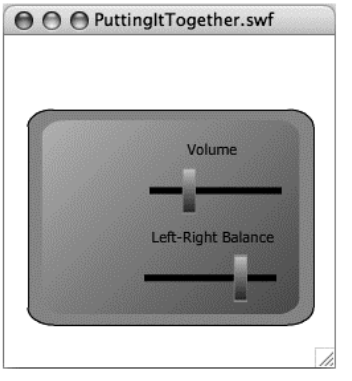
```
stage.addEventListener(Event.ENTER_FRAME, everyframe);
function everyframe(myevent:Event):void {
    var newvolume:Number = (volume_mc.x-volumegroove_mc.x)/100;
    var newbalance:Number = ((balance_mc.x-balancegroove_mc.x)/50)-1;
    newSetting.volume=newvolume;
    newSetting.pan=newbalance;
    myChannel.soundTransform=newSetting;
}
```

Rysunek 8.45. *W tej części skryptu dokonywane jest połączenie położeń suwaków z odpowiednimi własnościami dźwięku. W ramach funkcji wywoływanej przez zdarzenie Event.ENTER_FRAME dokonywane są pewne obliczenia, służące do uzyskania właściwego zakresu wartości: od 0 do 1 dla własności volume i od −1 do 1 dla własności pan. Potem te wartości są przypisywane do własności obiektu SoundTransform, a obiekt SoundTransform jest przypisywany do własności soundTransform obiektu SoundChannel*

8. Upewnij się, że Twój zewnętrzny plik MP3 znajduje się w miejscu, gdzie Flash może go odnaleźć, posługując się informacją zawartą w obiekcie `URLRequest`. Przetestuj film (*Sterowanie/Testuj film*).

Flash ładuje i odtwarza dźwięk. Możesz pociągnąć za suwak głośności lub suwak balansu podczas odtwarzania dźwięku, a natychmiast usłyszysz różnicę (rysunek 8.46).

```
//
// define rectangles for dragging constraints
//
var myvolumebounds:Rectangle=new Rectangle(volumegroove_mc.x, volumegroove_mc.y, volumegroove_mc.width, 1);
var mybalancebounds:Rectangle=new Rectangle(balancegroove_mc.x, balancegroove_mc.y, balancegroove_mc.width, 1);
//
//drag volume sliders
//
volume_mc.addEventListener(MouseEvent.MOUSE_DOWN, startdragvolume);
volume_mc.addEventListener(MouseEvent.MOUSE_UP, stopdrag);
balance_mc.addEventListener(MouseEvent.MOUSE_DOWN, startdragbalance);
balance_mc.addEventListener(MouseEvent.MOUSE_UP, stopdrag);
function startdragvolume(myevent:MouseEvent):void {
    myevent.target.startDrag(false, myvolumebounds);
}
function startdragbalance(myevent:MouseEvent):void {
    myevent.target.startDrag(false, mybalancebounds);
}
function stopdrag(myevent:MouseEvent):void {
    myevent.target.stopDrag();
}
//
// create objects
//
var myRequest:URLRequest=new URLRequest ("music.mp3");
var mySound:Sound = new Sound();
var myChannel:SoundChannel;
var newSetting:SoundTransform = new SoundTransform();
//
// load and play sound
//
mySound.load(myRequest);
myChannel=mySound.play();
//
// change volume and balance continuously
//
stage.addEventListener(Event.ENTER_FRAME, everyframe);
function everyframe(myevent:Event):void {
    var newvolume:Number = (volume_mc.x-volumegroove_mc.x)/100;
    var newbalance:Number = ((balance_mc.x-balancegroove_mc.x)/50)-1;
    newSetting.volume=newvolume;
    newSetting.pan=newbalance;
    myChannel.soundTransform=newSetting;
}
```



Rysunek 8.46. *Gotowy skrypt (po lewej) pozwala dynamicznie sterować głośnością i balansem odtwarzanego dźwięku za pośrednictwem pary suwaków (po prawej)*