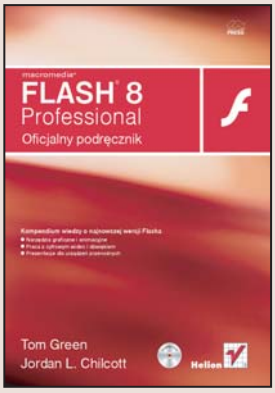
Macromedia Flash 8 Professional. Oficjalny podręcznik



Autorzy: Tom Green, Jordan L. Chilcott
Tłumaczenie: Paweł Kita
ISBN: 83-246-0313-1
Tytuł oryginału: Macromedia Flash 8 Professional: Training from the Source
Format: B5, stron: 480

Kompendium wiedzy o najnowszej wersji Flasha

- Narzędzia graficzne i animacyjne
- Praca z cyfrowym wideo i dźwiękiem
- Prezentacje dla urządzeń przenośnych

Macromedia Flash 8 Professional to najnowsza wersja aplikacji, która ciągle zmienia oblicze sieci WWW. Statyczne strony stają się tętniącymi życiem interaktywnymi prezentacjami. Stale rozwijany i udoskonalany język ActionScript umożliwia projektantom witryn WWW połączenie animacji utworzonych we Flashu z bazami danych i daje im niemal nieograniczoną kontrolę nad obiektami znajdującymi się na scenie. Wersja Professional oferuje dodatkowe możliwości obróbki cyfrowego wideo, tworzenia aplikacji w architekturze klient-serwer i publikowania prezentacji w formacie nadającym się do odtwarzania na wyświetlaczach urządzeń mobilnych.

„Macromedia Flash 8 Professional. Oficjalny podręcznik" to przygotowany przez specjalistów z firmy Macromedia zbiór ćwiczeń, które pozwolą Ci poznać możliwości tej wspaniałej aplikacji. Nauczysz się korzystania z narzędzi rysunkowych i tworzenia wektorowych animacji. Dowiesz się, jak za pomocą ActionScriptu kontrolować atrybuty obiektów oraz jak umieszczać w prezentacjach dźwięk i cyfrowe wideo. Poznasz sposoby wykorzystywania plików w formacie XML, łączenia aplikacji z Flash Communication Server i publikowania prezentacji.

- Interfejs użytkownika
- Podstawy języka ActionScript
- Tworzenie obiektów graficznych
- Elementy tekstowe
- Dołączanie dźwięku do prezentacji
- Animacja automatyczna i poklatkowa
- Praca z cyfrowym wideo
- Połączenie z serwerem komunikacyjnym
- Wykorzystywanie danych dynamicznych i plików XML
- Tworzenie prezentacji dla urządzeń mobilnych

Poznaj i wykorzystaj najlepsze koncepcje i techniki stosowane przy tworzeniu filmów Flasha

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

Spis treści

8 Tworzenie własnego odtwarzacza wideo w programie Flash

W lekcji 7. dowiedziałeś się, jak utworzyć odtwarzacz wideo za pomocą komponentu wideo. W tej lekcji opracujesz odtwarzacz wideo, który nie wykorzystuje komponentów. Możesz, oczywiście, zapytać: „Po co się męczyć, skoro mam dostęp do komponentów?".

Większość pracujących w programie Flash zawodowców unika wykorzystania komponentów, ponieważ zazwyczaj wiąże się to ze wzrostem rozmiaru pliku SWF. Dla przykładu odtwarzacz, który utworzyłeś w poprzednim ćwiczeniu, zajmował 35 kB, a do tego dochodziła jeszcze „skórka" o rozmiarze 15 kB. W niniejszej lekcji utworzysz odtwarzacz zawierający proste przyciski *Start* i *Stop*, którego wielkość nie przekracza 1 kB. Będzie się on znacznie szybciej wczytywał. Ponadto o wiele łatwiej będzie dostosować go do Twoich wymagań, ponieważ to Ty będziesz jego twórcą.

Pozostaje jeszcze kwestia elementów nawigacyjnych. Może się okazać, że potrzebujesz jedynie przycisków *Start* i *Pause*. W tym celu dodasz do sceny obiekt wideo i wykorzystasz te przyciski do kontroli odtwarzania, którą możesz uzyskać za pośrednictwem kodu ActionScript.

Nie potrzebujesz komponentów do kontroli odtwarzania wideo; możesz utworzyć własne elementy nawigacyjne

Lekcja została podzielona na trzy części. Pierwsza z nich pokazuje, w jaki sposób przesyła się materiał wideo za pomocą kodu ActionScript i obiektu wideo. W kolejnej części dowiesz się, jak dodawać przyciski, które umożliwiają użytkownikom kontrolę nad odtwarzaniem materiału wideo. Ostatnia część lekcji została poświęcona zastosowaniu kodu ActionScript do kontroli materiału wideo, który jest odtwarzany za pośrednictwem komponentu *FLVPlayback*. Składa się ona z dwóch projektów. Pierwszy z nich wyjaśnia proces odtwarzania wielu plików FLV naraz w celu utworzenia iluzji jednego dużego pliku wideo. Drugi projekt dotyczy dostępnej w kodzie ActionScript klasy `TransitionManager` — dowiesz się, jak sprawić, by kliknięcie przycisku spowodowało pojawienie się materiału wideo.

Czego dowiesz się w tym rozdziale?

W czasie tej lekcji:

- Utworzysz obiekt wideo na scenie programu Flash Professional 8,
- Wykorzystasz klasy NetConnection i NetStream do odtwarzania materiału wideo znajdującego się na serwerze,
- Dodasz do filmu elementy kontrolujące odtwarzanie,
- Wykorzystasz kod ActionScript do kontroli odtwarzania wielu plików FLV,
- Dowiesz się, w jaki sposób komponent FLVPlayback korzysta z akcji nasłuchujących w celu aktywowania zdarzeń,
- Poznasz dziesięć przejść, które wchodzą w skład klasy Transition,
- Dowiesz się, jak zarządzać tymi przejściami za pomocą klasy TransitionManager,
- Dowiesz się, jak uzyskać dostęp do przejść za pomocą słowa kluczowego Import.

Przybliżony czas trwania lekcji

Ukończenie tej lekcji zajmie około 90 minut.

Potrzebne pliki

Media:

Player.fla

Pliki początkowe:

Lekcja08/Complete/Player.fla

Gotowe pliki:

Converge.flv

Przesył strumieniowy materiału wideo

Musisz być świadomy tego, że przesył strumieniowy (zwłaszcza materiału wideo Flasha) oznacza wykorzystanie Twojego serwera sieciowego głównie w roli serwera strumieniowego. Masz także do dyspozycji inne rozwiązania, np. dostawcę Flash Video Streaming Server lub Flash Communication Server (o czym jest mowa w lekcji 10.), ale najpopularniejszą metodą przesyłu strumieniowego danych jest wykorzystanie serwera sieciowego w celu umożliwienia pobierania progresywnego. Kod ActionScript będzie taki sam, niezależnie od wybranej metody.

Wykorzystasz dwa rodzaje klas ActionScript. Pierwszy rodzaj związany jest z nawiązywaniem połączenia z (*connection*), klasa ta odpowiedzialna będzie za komunikację z serwerem. Na drugi rodzaj składają się klasy interfejsu użytkownika (*user interface*), a ich zadaniem będzie wyświetlanie filmu wideo w pliku SWF.

Wszystko to działa w bardzo prosty sposób. Plik zostanie załadowany do odtwarzacza Flash Player za pomocą klas `NetStream` i `NetConnection`, które z kolei prześlą materiał wideo do osadzonego na scenie obiektu wideo. Obiekt wykorzysta klasy `Video` i `Sound` do odtwarzania obrazu i kontroli głośności.

Klasa `NetConnection` jest odpowiedzialna za komunikację między serwerem a programem Flash. Klasa `NetStream` zarządza i kontroluje strumień, dzięki czemu użytkownik może odtwarzać, zatrzymywać, włączać pauzę, a nawet przewijać strumień wideo. Możesz wyobrazić sobie cały proces na przykładzie kabla, który znajduje się z tyłu telewizora. Kabel ten zapewnia połączenie z firmą udostępniającą usługi telewizyjne[1]. Programy przesyłane do telewizora za pośrednictwem kabla pełnią rolę strumienia.

Chociaż może to brzmieć trochę tajemniczo i nazbyt technicznie, poniżej przedstawiamy minimalną ilość kodu, której potrzeba do ustanowienia połączenia z serwerem i odtworzenia filmu wideo:

```
new NetConnection();
NetConnection.connect(null);
New NetStream(NetConnection);
NetStream.play("myVideo.flv");
Video.attachVideo(NetStream);
createEmptyMovieClip("newMovie",this.getNextHighestDepth());
newMovie.attachAudio(NetStream);
Sound soundObj = new Sound(newMovie);
```

Widać więc, że jest to trójstopniowy proces: *połącz, prześlij strumieniowo, odtwarzaj*. Kilka ostatnich wierszy kodu łączy strumień wideo z obiektem wideo i dźwiękiem filmu wideo.

Tworzenie obiektu wideo

Powodzenie przesyłu strumieniowego materiału wideo zależy od tego, czy plik FLV rozpocznie odtwarzanie zaraz po otwarciu strony internetowej. Dane pliku FLV przepłyną do pliku SWF za pośrednictwem `NetStream`. Dane zostaną następnie przesłane do znajdującego się na scenie obiektu wideo.

[1] Mówimy tu o telewizji kablowej — *przyp. tłum.*

Istnieje kilka ważnych zalet wykorzystania obiektu wideo.

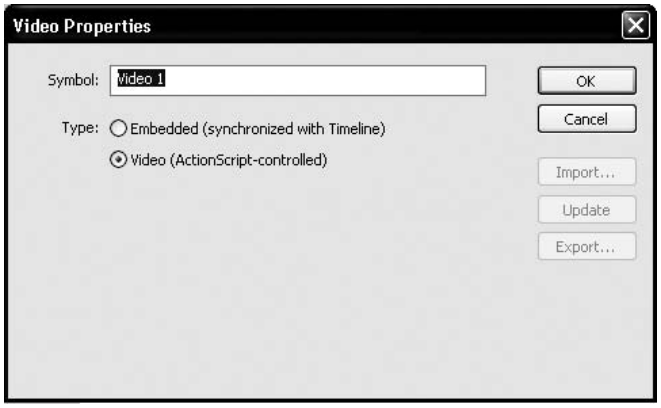
- **Nie musisz osadzać materiału wideo w pliku SWF**. W ten sposób unikniesz tworzenia wielkich plików SWF, co nie jest dobrym pomysłem, jeśli myśli się o widzach posiadających łącza typu dial-up.
- **Nie ma limitu klatek**. Materiał wideo osadzony w pliku SWF jest umieszczany na listwie czasowej. W zależności od długości filmu, może zdarzyć się, że przekroczysz limit listwy czasowej Flasha, który wynosi 16000 klatek. Najczęściej tworzy się odtwarzacze składające się z jednej klatki.
- **Wciśnięcie *Play* gwarantuje natychmiastowe odtwarzanie**. W momencie nawiązania połączenia i rozpoczęcia przesyłu strumieniowego rozpocznie się także odtwarzanie materiału wideo. W przypadku pobierania progresywnego pomiędzy połączeniem a odtwarzaniem wystąpi krótkie opóźnienie. Wykorzystanie serwera strumieniowego oznacza prawie natychmiastowe odtwarzanie.
- **Plik SWF jest bardzo mały**. W poprzedniej lekcji miałeś okazję przekonać się, że zastosowanie komponentów owocuje plikiem o rozmiarze około 70 kB. Wykorzystaj obiekt wideo i kilka przycisków, a rozmiar pliku zmniejszy się do poniżej 10 kB.
- **Nie ma ograniczeń długości filmów**. Długość filmu wideo, czy będzie to 1 minuta, czy też 10 minut, przy przesyłaniu strumieniowym nie ma znaczenia.

1. **Skopiuj na dysk twardy katalog *Lekcja08* z dołączonej do książki płyty CD. Otwórz plik *Player.fla*.**

   W czasie lekcji będziesz zapisywał i testował ten plik.

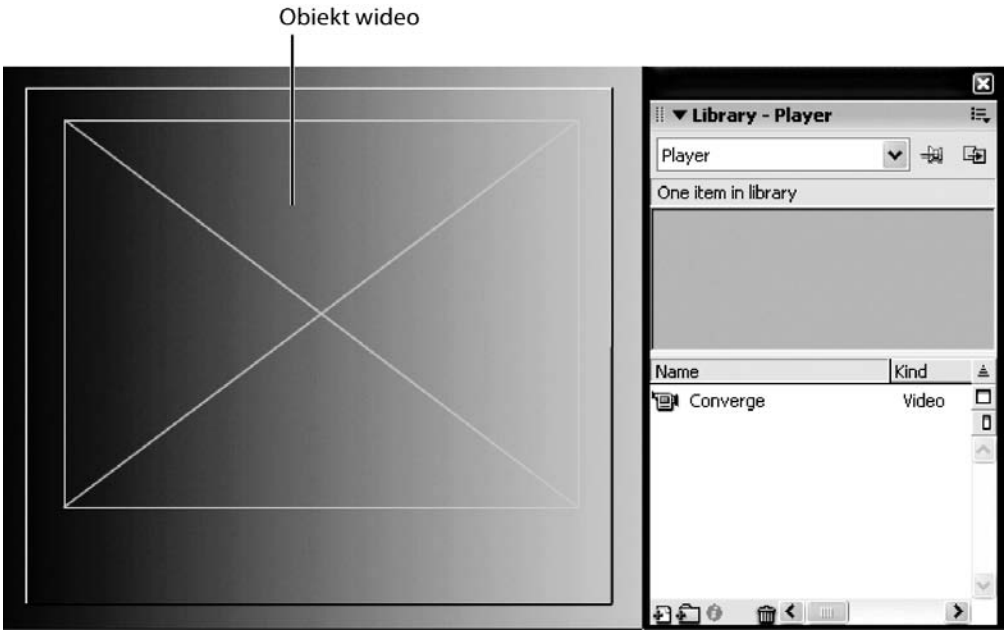
2. **Otwórz bibliotekę i wybierz opcję *New Video* z menu rozwijanego *Library Options*.**

   Tworzysz w ten sposób obiekt wideo, do którego zostanie przesłany plik FLV. Otworzy się okno dialogowe *Video Properties*, gdzie będziesz mógł wybrać rodzaj wideo — może to być znajdujący się bezpośrednio na listwie czasowej osadzony materiał wideo lub materiał zewnętrzny dołączony za pośrednictwem ActionScriptu.



3. **Po otwarciu okna dialogowego** *Video Properties* **nadaj symbolowi nazwę** `Converge` **i wybierz opcję** *Video (ActionScript-controlled)*. **Wciśnij** *OK*.

   W bibliotece pojawi się kamera z nazwą obiektu wideo. To jest właśnie obiekt wideo.



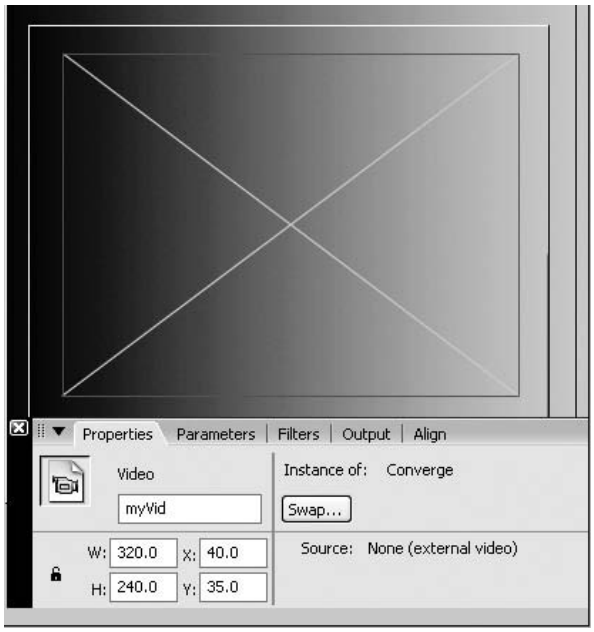
   Obiekt wideo określa na scenie miejsce, w którym zostanie wyświetlony film wideo. Będzie on niewidoczny do momentu odtworzenia materiału wideo w filmie Flasha. Cechą charakterystyczną obiektu wideo jest to, że korzysta on z klasy `Video`, która posiada metody i właściwości, ale nie zawiera zdarzeń. Obiekt ten może być skalowany, obracany, przesuwany, maskowany, a nawet umieszczany wewnątrz klipu filmowego. Nie kontroluje on natomiast odtwarzania materiału wideo. Działa podobnie jak telewizor — przechwytuje przychodzący strumień i wyświetla go. Elementy kontrolne zostaną dodane w dalszej części lekcji. Możesz także do obiektu wideo dołączyć dźwięk, ale rozwiązanie to jest rzadko spotykane, ponieważ większość plików wideo zawiera też ścieżkę dźwiękową.

4. **Zaznacz warstwę** *myVideo* **i przeciągnij na scenę kopię obiektu wideo**. **Nadaj mu nazwę instancji** `myVid` **w oknie** *Property inspector* **oraz ustaw wymiary** *H* **na** `240` **i** *W* **na** `320`, **a współrzędne** *x*,*y* **na** `40`,`35`. **Zapisz plik**.

   Obiekt wideo po umieszczeniu na scenie przybierze postać prostokąta z dużym „X” wewnątrz. Możesz także umieścić obiekt wideo w klipie filmowym, a nawet maskować go, ponieważ jest to obiekt znajdujący się na scenie (zobacz rysunek na następnej stronie).

Wskazówka

Nie musisz przesyłać strumieniowo pliku FLV do obiektu wideo. Możesz podłączyć do komputera kamerę internetową i przesyłać sygnał wideo z kamery do obiektu wideo.



Obiekt wideo zostanie dodany do biblioteki. Otrzyma nazwę instancji, zmieni rozmiar i zostanie umieszczony na scenie za pośrednictwem okna *Property inspector*.

Łączenie odtwarzacza Flash Player 8 z serwerem sieciowym

Za proces połączenia odpowiedzialna jest klasa `NetConnection`. Po dodaniu kodu do filmu, który zawiera obiekt wideo, klasa `NetConnection` zajmie się komunikacją, zachodzącą między komputerem użytkownika a serwerem. Wprowadzona w odtwarzaczu Flash Player 6 klasa `NetConnection` może na jednym połączeniu obsługiwać wiele strumieni danych.

Klasa `NetConnection` zawiera trzy metody, z których dwie dostępne są tylko wtedy, gdy używasz Flash Communication Server lub Flash Video Streaming Service. Oto one:

- **`NetConnection.connect()`:** żądanie ustanowienia połączenia z serwerem, na którym znajduje się plik FLV,
- **`NetConnection.close()`:** metoda wykorzystywana tylko w przypadku serwerów strumieniowych — zamyka połączenia z serwerem,
- **`NetConnection.call()`:** metoda wykorzystywana tylko w przypadku serwerów strumieniowych — może być wykorzystana w celu odwołania się do metod odtwarzacza Flash Player.

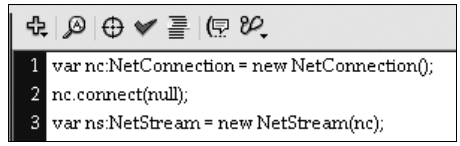
Po ustanowieniu połączenia można rozpocząć przesył strumieniowy — następnie można połączyć klasę `NetStream` i obiekt wideo ze strumieniem.

1. **Zaznacz warstwę *Actions* i wciśnij *F9*, by otworzyć edytor ActionScriptu**.

2. **Wpisz poniższy kod w panelu *Actions***:

```
var nc:NetConnection = new NetConnection();
nc.connect(null);
```

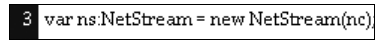
Pierwszy wiersz powyższego kodu definiuje obiekt połączenia i nadaje mu nazwę instancji (`nc`). W drugim wierszu znajduje się metoda `connect`, służąca do ustanawiania połączenia. Parametr `null` ustala połączenie między lokalnym serwerem sieciowym a dyskiem twardym.



3. **Wciśnij *Return* lub *Enter* i wpisz poniższy kod**:

```
var ns:NetStream = new NetStream(nc);
```

Ten wiersz definiuje obiekt `NetStream` i tworzy jego instancję wykorzystując jako parametr zdefiniowany wcześniej obiekt `NetConnection`. Filmy Flasha muszą zawierać zarówno obiekt `NetStream`, jak i `NetConnection` w celu skierowania strumienia danych do filmu Flasha.



Uwaga

Dane audio i wideo są dołączane do strumienia w celu przesłania ich do komputera użytkownika. Film, który tworzy strumień w połączeniu sieciowym i wykorzystuje go do wysyłania danych do użytkowników, określa się mianem nadawcy (publisher) — oznacza to, że publikuje on strumień. Film, który tworzy strumień w celu odbierania danych (przykładem tego jest niniejsza lekcja), określany jest mianem odbiorcy (subscriber). Musisz pamiętać, że strumień może zawierać tylko jeden strumień audio i wideo naraz. Pięć filmów wideo nie może korzystać z jednego strumienia. Gdy chcesz odtworzyć inny film, musisz usunąć ze strumienia materiał, który się w nim znajduje, i zastąpić go nowym lub też utworzyć nowy strumień, do którego dołączysz obiekt wideo.

4. **Wciśnij *Enter* lub *Return* i wpisz poniższy kod**:

```
var myVid: Video;
myVid.attachVideo(ns);
ns.play("Converge.flv");
```

Po ustanowieniu `NetConnection` i `NetStream` nadajemy obiektowi wideo nazwę `myVid`, która jest taka sama jak nazwa instancji znajdującego się na scenie obiektu. Obiekt wideo jest następnie dołączany do obiektu `NetStream` za pomocą metody `attachVideo()`. Strumień i obiekt wideo są już połączone, więc ostatni wiersz kodu wykorzystuje metodę `play()` w celu wczytania pliku *Converge.flv* do `NetStream` i przesłania go do obiektu wideo.

Uwaga

Jeśli korzystasz z podpowiedzi kod (Code Hinting) przy wyborze typów danych dla zmiennych, musisz wiedzieć, że klasa wideo nie znajduje się w menu rozwijanym. Wykorzystanie tej klasy jest ograniczone wyłącznie do obiektów wideo, co tłumaczy pominięcie jej w tej liście.

```
1 var nc:NetConnection = new NetConnection();
2 nc.connect(null);
3 var ns:NetStream = new NetStream(nc);
4 var myVid: Video;
5 myVid.attachVideo(ns);
6 ns.play("Converge.flv");
```

5. Zapisz plik i przetestuj go.

Rozpocznie się odtwarzanie filmu wideo w ramach znajdującego się na scenie obiektu wideo.

Nie zamykaj pliku — wykorzystasz go w kolejnym ćwiczeniu.

Uwaga

Powyższy kod nie został wprowadzony według typowych konwencji zapisu, według których zmienne deklaruje się na samym początku. Zdecydowaliśmy się na deklarację zmiennych w odpowiednich wierszach dlatego, że chcemy, byś wiedział, jak działa program. Według konwencji zapisu ten kod powinien wyglądać w następujący sposób:

```
var nc:NetConnection = new NetConnection();
var ns:NetStream = new NetStream(nc);
var myVid: Video;
nc.connect(null);
myVid.attachVideo(ns);
ns.play("Converge.flv");
```

Dodawanie elementów kontrolujących odtwarzanie

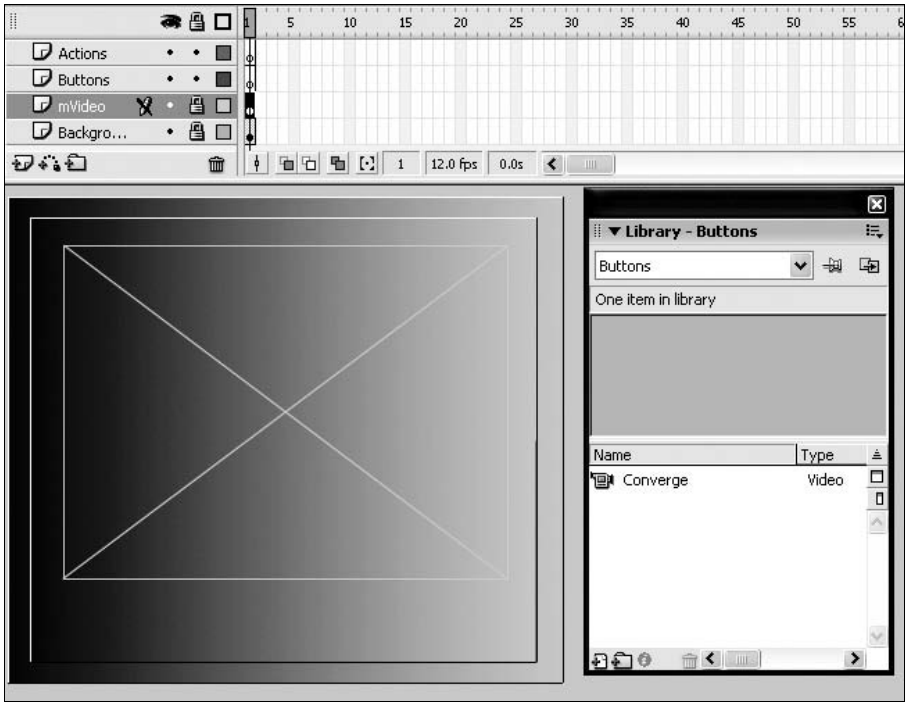
Komponenty, które znajdują się w panelu *FLV Custom UI-Flash 8*, nie są kompatybilne z obiektem wideo. Może się to na początku wydawać wadą, ale w rzeczywistości jest to zaleta, ponieważ komponenty zwiększają rozmiar pliku SWF. Utwórz własne elementy nawigacyjne, a zauważysz, że rozmiar pliku SWF jest mniejszy.

W tym ćwiczeniu wykorzystasz kilka gotowych przycisków, które zostały dołączone do programu Flash Professional 8. Jeśli uznasz, że nie pasują one do Twojego projektu, możesz zawsze utworzyć własne w programach, takich jak Illustrator CS2 lub Photoshop CS2 firmy Adobe lub Freehand MX i Fireworks 8 firmy Macromedia. Jeśli się postarasz, możesz je nawet przygotować w programie Flash Professional 8.

Pamiętaj, że przyciski nie kontrolują filmu wideo. Ich zadaniem jest kontrola `NetStream`. Kliknięcie przycisku *Pause* nie oznacza zatrzymania filmu wideo, a wysłanie wiadomości do serwera, by ten zatrzymał wysyłanie.

1. **Wykorzystasz plik, który zapisałeś w poprzednim ćwiczeniu**.

   Jeśli nie zapisałeś tego pliku, otwórz *Buttons.fla*, który znajduje się w katalogu *Buttons* w *Lekcji 8*. Dodaj nową warstwę i nazwij ją `buttons`.



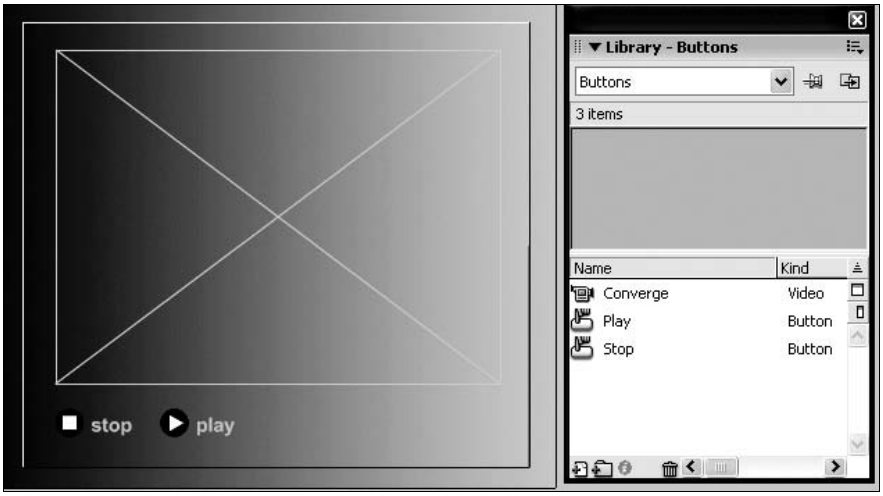
2. **Wybierz** *Window/Common Libraries/Buttons*.

Otworzy się panel *Library-Buttons*. Obie wersje Flasha — Flash 8 i Flash Professional 8 — zawierają duży wybór przycisków, które mogą być stosowane do tak różnych celów jak nawigacja i kontrola materiału wideo.

Przyciski te znajdują się w kilku katalogach biblioteki *Buttons*. Dodanie do sceny przycisku z tego panelu sprawia, że możesz robić z nim, co zechcesz, a oryginał znajdujący się w bibliotece pozostanie niezmieniony.

3. **Otwórz folder** *Circle Buttons*, **który znajduje się w katalogu** *classic buttons* **w panelu** *Library-Buttons*. **Zaznacz warstwę** *buttons* **i przeciągnij na scenę kopie przycisków** *Play* **i** *Stop*.

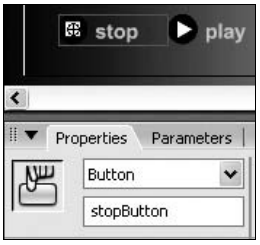
Przeciągnięcie na scenę przycisku z panelu *Buttons* sprawi, że w bibliotece także pojawi się kopia przycisku.



4. Zaznacz przycisk *Play* i w oknie *Property inspector* nadaj mu nazwę instancji `playButton`.

5. Zaznacz przycisk *Stop* i w oknie *Property inspector* nadaj mu nazwę instancji `stopButton`.

Przyciski mają już nazwy instancji, więc można użyć ich wraz z kodem ActionScript do kontroli odtwarzania wideo.

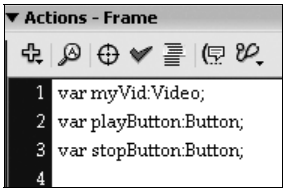


6. Zaznacz 1. klatkę warstwy *Actions* i wciśnij *F9*, by otworzyć panel *Actions*.

Usuń całość kodu, który wpisałeś w poprzednim ćwiczeniu, i zastąp go poniższym, zaczynając od wiersza 1.:

```
var myVid:Video;
var playButton:Button;
var stopButton:Button;
```

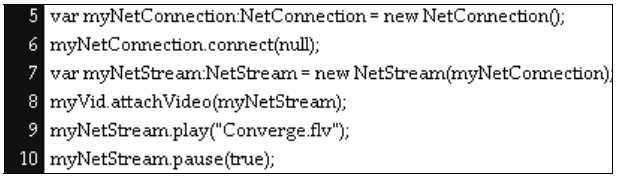
Deklarując zmienne i przypisując im typy danych, upewniasz się, że można do nich zastosować tylko taki kod, który odnosi się do materiału wideo lub przycisków związanych z tymi zmiennymi.




```
var myNetConnection:NetConnection = new NetConnection();
myNetConnection.connect(null);
var myNetStream:NetStream = new NetStream(myNetConnection);
myVid.attachVideo(myNetStream);
myNetStream.play("Converge.flv");
myNetStream.pause(true);
```

Ostatni wiersz kodu — `myNetStream.pause(true);` — wyłącza strumień i wstrzymuje odtwarzanie do momentu, w którym użytkownik wciśnie przycisk *Play*. Zauważ, że wartość logiczna tego stwierdzenia to `true`, czyli prawda.

Wszystko jest kwestią kontroli. Wykorzystanie pauzy bez parametru oznacza po prostu przełączanie pomiędzy stanem pauzy a stanem odtwarzania. W tym przypadku nie mamy do czynienia z prawdziwą kontrolą. Tę kontrolę zapewnia wartość logiczna. Wartość `true` oznacza, że wiemy, iż chcemy włączyć pauzę podczas odtwarzania wideo. Analogicznie wartość `false` oznacza chęć rozpoczęcia odtwarzania. Wykorzystanie przełącznika pauzy sprawdza się w przypadku pojedynczego przycisku, który może włączać i wyłączać odtwarzanie. Jeśli jednak chcesz wstrzymywać odtwarzanie w sposób świadomy, powinieneś użyć wartości logicznej.



7. Wciśnij *Return* lub *Enter* i wprowadź poniższy kod:

```
playButton.onPress = function() {
      myNetStream.pause(false);
}

stopButton.onPress= function() {
      myNetStream.pause(true);
}
```

Kliknięcie przycisku spowoduje wywołanie metody `pause(true);` do zatrzymania strumienia oraz `pause(false);` do rozpoczęcia go.

```
1  var myVid:Video;
2  var playButton:Button;
3  var stopButton:Button;
4
5  var myNetConnection:NetConnection = new NetConnection();
6  myNetConnection.connect(null);
7  var myNetStream:NetStream = new NetStream(myNetConnection);
8  myVid.attachVideo(myNetStream);
9  myNetStream.play("Converge.flv");
10 myNetStream.pause(true);
11
12 playButton.onPress = function() {
13       myNetStream.pause(false);
14 }
15
16 stopButton.onPress= function() {
17       myNetStream.pause(true);
18 }
```

8. Zapisz film i przetestuj go.

Po rozpoczęciu filmu ekran będzie pusty. Kliknij przycisk *Play*, a rozpocznie się odtwarzanie materiału wideo. Zatrzymaj je przyciskiem *Pause*, a następnie znowu wciśnij *Play*, by ponownie włączyć odtwarzanie.

Odtwarzanie wielu filmów wideo

Ćwiczenia zawarte w rozdziałach tym i poprzednim koncentrowały się na odtwarzaniu pojedynczego filmu wideo. Bardzo ciekawym aspektem komponentu *FLVPlayback* programu Flash Professional 8 jest możliwość tworzenia wirtualnej playlisty składającej się z filmów wideo, które

będą odtwarzane automatycznie. Jest to bardzo ważna technika i warto ją poznać. Zamiast przesyłać strumieniowo do odtwarzacza jeden „zbyt duży" plik FLV, możesz odtworzyć serię krótszych filmów. Ponadto playlista umożliwia dodawanie, usuwanie, a nawet zamienianie filmów bez konieczności poświęcania czasu na przygotowanie pliku MOV, konwertowanie go do pliku FLV, a następnie wczytywanie do odtwarzacza.

W poniższym ćwiczeniu zbudujesz odtwarzacz, który może wyświetlać dwa filmy — *Warrior.flv* i *MAMMAAND.flv*. Umieszcza on także tytuł wyświetlanego filmu w polu tekstowym, które znajduje się pod obrazem. Efekt ten uzyskujemy dzięki utworzeniu listy plików FLV, które mają zostać wykorzystane, a następnie wczytaniu materiału wideo do instancji komponentu wczytanego w razie konieczności.

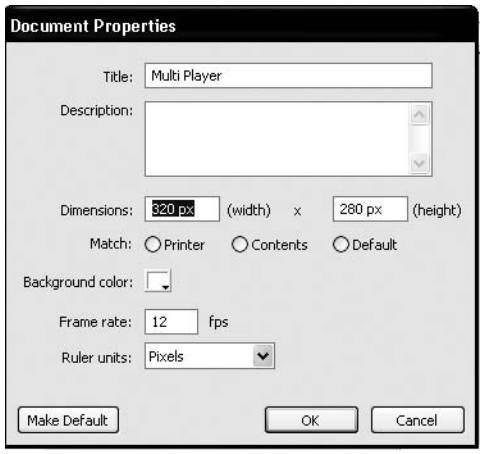
Filmy odtwarzane będą jeden po drugim dzięki zastosowaniu właściwości pliku FLV — czasu trwania. Napiszesz krótką funkcję, która po zakończeniu odtwarzania bieżącego filmu rozpocznie odtwarzanie następnego w kolejce. Dodasz też jeszcze jedną, krótką funkcję, która będzie stale sprawdzała kolejność filmów, a po zakończeniu odtwarzania ostatniego elementu rozpocznie cały proces od początku. Jest to możliwe dzięki właściwości listy, która nosi nazwę `length` (długość). Po dotarciu do ostatniego elementu listy film zostanie odtworzony do końca, a następnie proces zatrzyma się.

Wskazówka

W poniższym ćwiczeniu wykorzystamy tylko dwa filmy, ale musisz pamiętać, że równie łatwo byłoby utworzyć odtwarzacz zawierający 10 filmów. Musisz tylko wpisać w kodzie nazwy plików wideo.

1. **Uruchom program Flash Professional 8, utwórz nowy dokument i w oknie** *Property inspector* **ustaw rozmiary sceny na** `320×280`.

   Materiał wideo ma wymiary 320×240, ale musisz także zostawić miejsce na tytuł wyświetlanego filmu.



2. **Dodaj dwie warstwy do listwy czasowej. Nadaj istniejącym warstwom nazwy** `Actions`**,** `Video` **i** `Text`**.**

   Umieszczenie poszczególnych elementów w oddzielnych warstwach sprawi, że będziesz mógł je łatwiej zaznaczać.

3. **Zaznacz warstwę** *Text*, **wybierz narzędzie** *Text Tool* **następnie kliknij i przeciągnij, by utworzyć na scenie pole tekstowe. Wpisz do tego pola kilka liter wybranych losowo.**

   W oknie *Property inspector* wprowadź poniższe dane:

   - *Text type: Dynamic text*
   - *X:* `23`
   - *Y:* `250`
   - *Instance name:* `vidName`
   - *Font: Arial*
   - *Size:* `16`
   - *Color:* czarny (`#000000`)
   - *Styl:* pogrubienie
   - *Aliasing: Anti-alias for Readability*

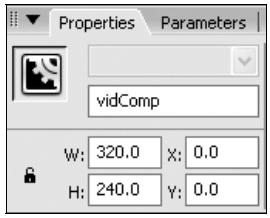
Wskazówka

Losowy ciąg liter ma pozwolić Ci na obejrzenie tekstu, który pojawi się w polu tekstowym podczas odtwarzania filmu. Zostanie on zastąpiony tytułem odtwarzanego filmu wideo.

4. Zaznacz warstwę *Video* i przeciągnij instancję komponentu *FLVPlayback* na scenę.

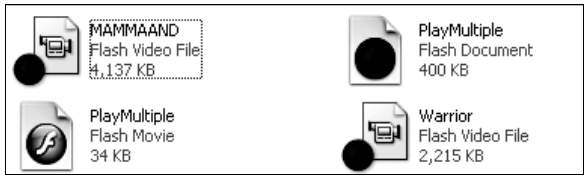
Upewnij się, że komponent jest zaznaczony i w oknie *Property inspector* wpisz poniższe wartości:

- *Width:* `320`
- *Height:* `240`
- *X:* `0`
- *Y:* `0`
- *Instance name*: `vidComp`



5. Zapisz plik w folderze *Lekcja08*.

Pamiętaj, by zapisać pliki SWF i FLA w katalogu, w którym znajdują się wykorzystywane pliki FLV. Filmy wideo zostaną wczytane z katalogu, w którym znajduje się plik FLA. Jeśli w katalogu tym nie ma materiału wideo, filmy nie zostaną odtworzone.

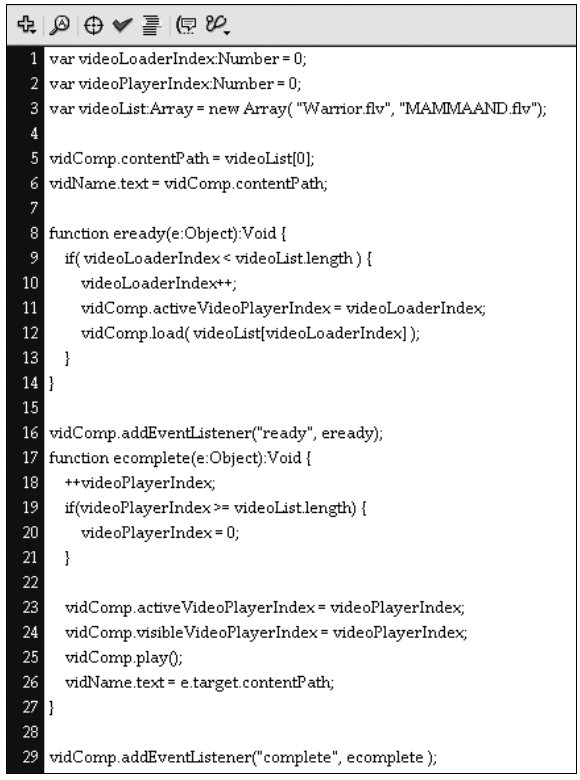


6. Zaznacz warstwę *Actions* i wciśnij *F9*, by otworzyć panel *Actions*.

W panelu *Actions* wpisz poniższy kod:

```
var videoLoaderIndex:Number = 0;
var videoPlayerIndex:Number = 0;
var videoList:Array = new Array("Warrior.flv", "MAMMAAND.flv");
vidComp.contentPath = videoList[0];
vidName.text = vidComp.contentPath;

function eready(e:Object):Void {
   if( videoLoaderIndex < videoList.length ) {
      videoLoaderIndex++;
      vidComp.activeVideoPlayerIndex = videoLoaderIndex;
      vidComp.load( videoList[videoLoaderIndex] );
   }
}
vidComp.addEventListener("ready", eready);
function ecomplete(e:Object):Void {
   ++videoPlayerIndex;
   if(videoPlayerIndex >= videoList.length) {
      videoPlayerIndex = 0;
}
   vidComp.activeVideoPlayerIndex = videoPlayerIndex;
   vidComp.visibleVideoPlayerIndex = videoPlayerIndex;
   vidComp.play();
   vidName.text = e.target.contentPath;
}
vidComp.addEventListener("complete", ecomplete );
```



Pierwsza część kodu ustawia początkowe wartości listy wideo.

```
var videoLoaderIndex:Number = 0;
var videoPlayerIndex:Number = 0;
var videoList:Array = new Array( "Warrior.flv", "MAMMAAND.flv");
```

Pierwsza zmienna (`videoLoaderIndex`) ma za zadanie przechowywać pierwszą wartość listy wideo, która następnie wczyta filmy do komponentu *FLVPlayback*. Druga zmienna ma w zasadzie tę samą funkcję, ale jej wartość zostanie użyta, by wskazać komponentowi, który film wideo należy odtworzyć. Trzecia zmienna utworzy listę filmów wideo. Właśnie w tym miejscu powinieneś wpisać nazwy plików FLV, które mają zostać odtworzone.

Kolejne dwa wiersze mają za zadanie wczytać pierwszy film wideo z listy do komponentu, a ponadto dodać nazwę pliku do dynamicznego pola tekstowego, które znajduje się na scenie:

```
vidComp.contentPath = videoList[0];
vidName.text = vidComp.contentPath;
```

W dalszej części kodu znajdują się dwie funkcje, które wykonują całą pracę. Pierwsza z nich tworzy instancję odtwarzacza wideo dla każdego filmu, który znajduje się na liście wideo wewnątrz komponentu *FLVPlayback* (lista jest następnie wykorzystywana do wczytania filmu wideo do instancji):

```
function eready(e:Object):Void {
   if( videoLoaderIndex < videoList.length ) {
      videoLoaderIndex++;
      vidComp.activeVideoPlayerIndex = videoLoaderIndex;
      vidComp.load( videoList[videoLoaderIndex] );
   }
}
```

Pierwszy wiersz ma na celu zapewnienie, że wczytanie każdego filmu wideo jest powiązane ze zdarzeniem `ready`, które jest aktywowane przez komponent za każdym razem, gdy wczytywany jest materiał wideo. Oczywiście, liczba filmów nie jest nieskończona, więc instrukcja warunkowa działa w następujący sposób:

- Sprawdza, czy wszystkie filmy wideo zostały wczytane. Jeśli tak, to wszystko gotowe. Jeśli są jeszcze jakieś filmy, zostanie wykonany kod znajdujący się w bloku instrukcji warunkowej `if(videoLoaderIndex < videoList.length)`.
- Inkrementuje wskaźnik indeksujący, by wskazywał następny film, który znajduje się w tablicy: `videoList(videoLoaderIndex++;)`.

Dwa kolejne wiersze funkcji wskazują programowi Flash, by ustawił `activeVideoPlayerIndex` poprzez wskazanie komponentowi *FLVPlayback* (`vidComp`), który odtwarzacz wideo zostanie użyty do wczytania filmu:

```
vidComp.activeVideoPlayerIndex = videoLoaderIndex;
vidComp.load( videoList[videoLoaderIndex] );
```

Po zidentyfikowaniu filmu kolejny wiersz kodu wczyta go do odtwarzacza, ale go nie uruchomi.

Następny wiersz łączy właśnie utworzoną funkcję z obiektem nasłuchującym komponentu *FLVPlayback*:

```
vidComp.addEventListener("ready", eready);
```

Program Flash wie już, jakie filmy znajdują się w której instancji komponentu — pozostaje jeszcze wskazać mu, co powinien zrobić po zakończeniu odtwarzania filmu. Właśnie tym zajmuje się druga funkcja. Jest ona aktywowana za każdym razem, gdy kończy się film.

Pierwszy wiersz funkcji rozkazuje programowi Flash przejście do następnego filmu:

```
++videoPlayerIndex;
```

Następny wiersz sprawdza listę wartości, by dowiedzieć się, czy nie przekroczyłeś liczby filmów, które mają być odtworzone; jeśli tak, proces rozpocznie się od nowa poprzez wyzerowanie wartości `videoPlayerIndex`. Jeśli wartość jest niższa, wiersz ten zostanie pominięty.

Pozostałe wiersze funkcji napędzają to, co widzisz na ekranie:

```
vidComp.activeVideoPlayerIndex = videoPlayerIndex;
vidComp.visibleVideoPlayerIndex = videoPlayerIndex;
vidComp.play();
vidName.text = e.target.contentPath;
```

Właściwość `activeVideoPlayerIndex` dostaje informację o tym, którego filmu z listy użyć, następnie informacja ta przekazywana jest również do właściwości `visibleVideoPlayerIndex`, która określa, który film ma być widoczny, Gdy odtwarzacz stanie się widoczny, rozpocznie się odtwarzanie filmu, a jego nazwa pojawi się w polu tekstowym.

Ostatni wiersz łączy funkcję z komponentami poprzez dopisanie jej do obiektu nasłuchującego (`addEventListener()`); funkcja ta zostanie wywołana po zakończeniu odtwarzania filmu:

```
vidComp.addEventListener("complete", ecomplete);
```

Komponenty wpływają na zdarzenia w inny sposób niż klipy filmowe i przyciski. Klipy filmowe i przyciski reagują na uchwyty zdarzeń, np. `onPress`. Komponenty wykorzystują obiekty nasłuchujące.

Model zdarzeń oparty o obiekty nasłuchujące składa się z dwóch części: nadawcy i słuchacza[2]. Aby łatwiej zrozumieć tę różnicę, pomyśl o rozmowie telefonicznej. Uchwyt zdarzenia klipu filmowego lub przycisku to osoba, która mówi do telefonu. W takiej rozmowie są tylko dwie strony i tylko jedna z nich słyszy głos drugiej. Zdarzenie oparte o obiekt nasłuchujący przypomina raczej audycję radiową, w której może uczestniczyć np. 200 słuchaczy zamiast dwóch.

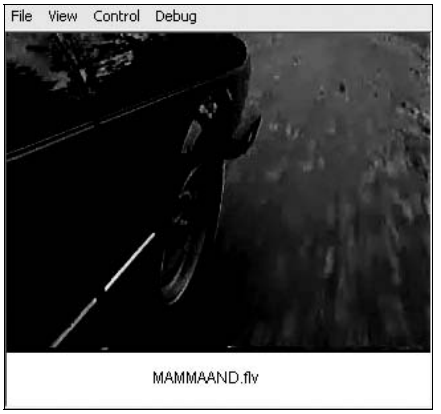
Obiekty nasłuchujące zapewniają użytkownikom wielką elastyczność, a ich zalety opisaliśmy poniżej.

[2] Nazwę „listener" można przetłumaczyć właśnie jako słuchacz, ktoś kto nasłuchuje — *przyp. tłum.*

- **Do jednego zdarzenia można przypisać nieskończoną liczbę obiektów nasłuchujących**. Dla ważnego zdarzenia możesz aktywować dowolną liczbę uchwytów zdarzeń. W tym ćwiczeniu zdarzeniem jest zakończenie filmu. Komponent nasłuchuje w poszukiwaniu tego zdarzenia, po czym znika.
- **Dla wielu zdarzeń można ustawić obiekty nasłuchujące tak, by reagowały tylko na niektóre z nich**. Jeśli jakaś radiostacja transmituje piosenkę, którą lubisz, będziesz jej słuchał. Jeśli jednak będzie to piosenka, której nie lubisz, po prostu ją zignorujesz. Obiekty nasłuchujące działają w ten sam sposób. W przypadku naszych filmów, gdy zakończy się odtwarzanie, wyłączy się tylko komponent odtwarzający materiał wideo. Zupełnie nie interesuje go to, że kolejny wczytuje się i przygotowuje do odtwarzania.
- **Obiekt nasłuchujący może reagować na wiele zdarzeń**. Przyciski reagują tylko na zdarzenie `onPress`. Obiekt nasłuchujący może zareagować na dowolne zdarzenie (np. koniec filmu), ponieważ wszystkie zdarzenia są dla niego dostępne.

7. **Zapisz film i przetestuj go**.

Zauważ, że oba filmy wyglądają tak, jakby były jednym plikiem FLV.



Wykorzystanie klasy TransitionManager w celu dodania do filmu przejścia Fade

W poprzednim ćwiczeniu dowiedziałeś się, jak wykorzystać kod ActionScript do odtworzenia wielu filmów w taki sposób, by wyglądały jak jeden duży film. Poniższe ćwiczenie oparte jest na wiedzy, którą już posiadasz. Zamiast jednak tworzyć jeden duży film, utworzysz film, który:

- odtwarza film po wciśnięciu odpowiedniego przycisku,
- wykorzystuje przejście *Fade* po wciśnięciu przycisku uruchamiającego drugi film.

Wykorzystanie klasy `Transition` (przejście) powoduje, że film, który jest połączony z przyciskiem, zacznie pojawiać się ponad odtwarzanym materiałem wideo.

Podczas instalacji programu Flash Professional 8 zainstalowałeś także dwie bardzo ważne klasy, `Tween` i `Transition`. Dzięki nim za pośrednictwem kodu ActionScript możesz dodawać efekty specjalne do klipów filmowych i komponentów. W poniższym ćwiczeniu utworzysz trzy instancje komponentu *FLVPlayback* i wykorzystasz przejście *Fade*, by sprawić, że materiał wideo będzie pojawiał się i znikał, w zależności od wybranego przycisku.

Program Flash Professional 8 zawiera 10 przejść, których opis znajdziesz poniżej.

- **Iris**. Odsłania ekran lub klip filmowy za pomocą maski, która wykonuje zbliżenie.
- **Wipe**. Odsłania ekran lub klip filmowy za pomocą animowanej maski o kształcie, który porusza się w poziomie.
- **Pixel Dissolve**. Ekran lub klip filmowy są maskowane za pomocą prostokątów, które pojawiają się i znikają.
- **Blinds**. Odsłania następny ekran lub klip filmowy za pomocą prostokątów, które pojawiają się i znikają.
- **Fade**. Ekran lub klip filmowy powoli pojawiają się lub zanikają.
- **Fly**. Ekran lub klip filmowy nasuwają się z określonego kierunku.
- **Zoom**. Przybliżenie lub oddalenie ekranu albo klipu filmowego.
- **Squeeze**. Zmiana rozmiaru ekranu lub klipu filmowego w pionie lub poziomie.
- **Rotate**. Obrót ekranu lub klipu filmowego.
- **Photo**. Ekran lub klip filmowy wyglądają tak, jakby zostały oświetlone lampą aparatu fotograficznego.

Powyższe przejścia są dostępne za pośrednictwem klasy `TransitionManager`, której funkcją jest zarządzanie przejściami. Umożliwia ona przypisanie jednego z przejść do klipów filmowych lub komponentów za pomocą kodu ActionScript.

1. **Uruchom program Flash Professional 8, utwórz nowy dokument i w oknie** *Property inspector* **ustaw wymiary sceny na** `320×280`.

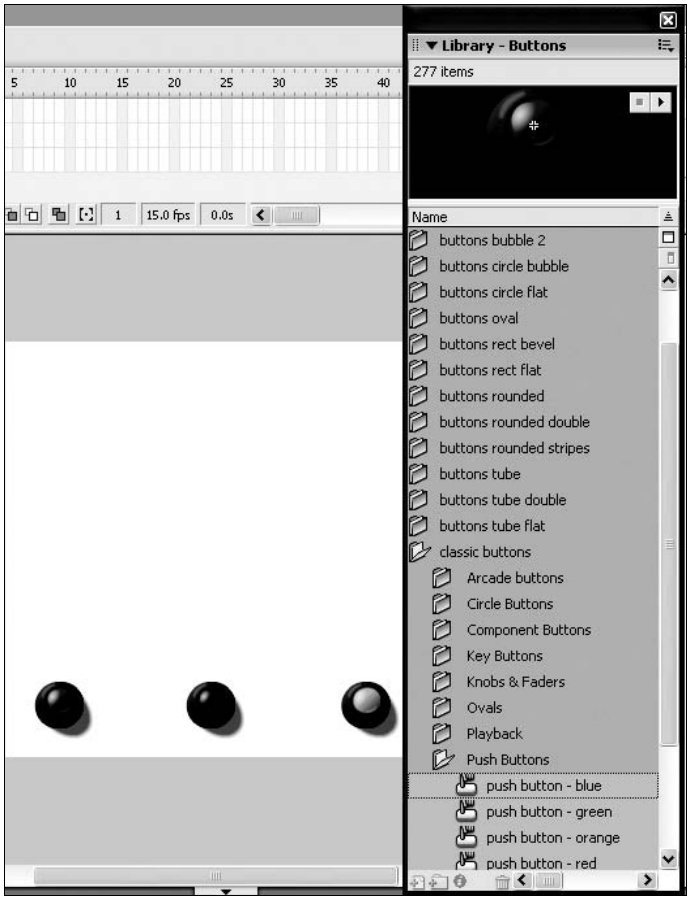
   Filmy, których użyjesz, mają wymiary `320×240`, ale musisz zostawić też miejsce na przyciski.

2. **Dodaj dwie nowe warstwy i nadaj im nazwy** `Actions`, `Video` **i** `Buttons`.

   Rozdzielenie elementów umożliwia łatwiejsze ich zaznaczanie.

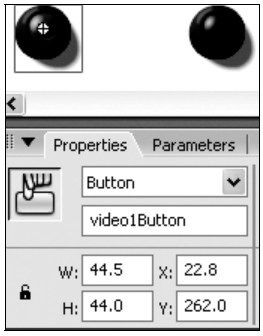
3. **Zaznacz warstwę** *Buttons* **i wybierz** *Window/Common Libraries/Buttons*.

   Otwórz katalog *classic buttons/Push Buttons* w panelu *Library-Buttons*. Przeciągnij kopie *push button-yellow*, *push button-blue* oraz *push button-red* na scenę (zobacz rysunek na następnej stronie).



4. **Zaznacz każdy z przeniesionych na scenę przycisków i w oknie** *Property inspector* **nadaj im nazwy instancji:**

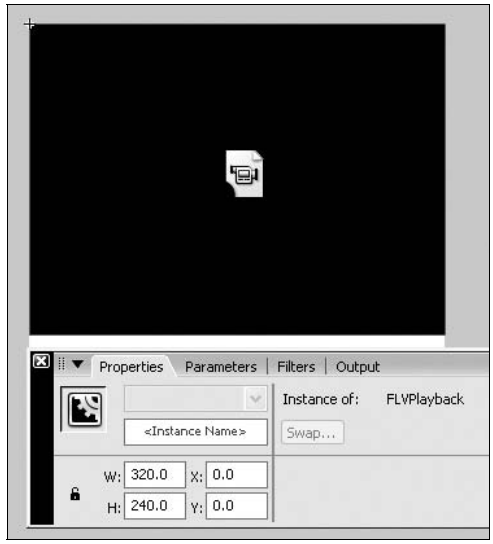
   - `video1Button`
   - `video2Button`
   - `video3Button`



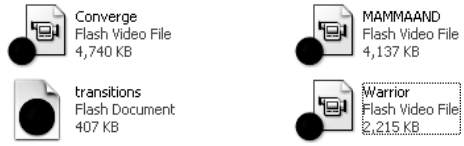
5. Zaznacz warstwę *Video* i przeciągnij na scenę instancję komponentu *FLVPlayback*.

Zaznacz ten komponent i w oknie *Property inspector* wpisz poniższe wartości:

- *Width:* `320`
- *Height:* `240`
- *X:* `0`
- *Y:* `0`
- *Instance name:* `vidcomp`



6. Zapisz plik w katalogu *Lekcja08*.



7. Zaznacz warstwę *Actions* i wciśnij *F9*, by otworzyć panel ActionScript.

W panelu *Actions* wpisz poniższy kod:

```
import mx.transitions.*;

var videoLoadedCount:Number = 0;
var videoPlayerIndex:Number = 0;
var videoList:Array = new Array("Warrior.flv","Converge.flv",
"MAMMAAND.flv");
for (var loaderLoop:Number = 1; loaderLoop <= videoList.length; loaderLoop++
) {
      vidcomp.activeVideoPlayerIndex = loaderLoop;
      vidcomp.load( videoList[loaderLoop - 1] );
```

```
}
function eready(e:Object):Void {
   ++videoLoadedCount;
   if ( videoLoadedCount == videoList.length ) {
   for (var playerLoop:Number = 1; playerLoop <= videoList.length;
   playerLoop++ ) {
         e.target.activeVideoPlayerIndex = playerLoop;
         e.target.play();
      }

   }
}

vidcomp.addEventListener("ready", eready);
function ecomplete(e:Object):Void {
   for (var playerLoop:Number = 1; playerLoop <= videoList.length;
   playerLoop++ ) {
      e.target.activeVideoPlayerIndex = playerLoop;
      e.target.play();
      }
}
vidcomp.addEventListener("complete", ecomplete);
function transDone(e) {
   vidcomp.visibleVideoPlayerIndex = e.target.content._name;
   trace(e.target.content._name);
}

function buttonTransition(m:MovieClip, d:Number) {
   if (d != m.visibleVideoPlayerIndex) {
      var other:MovieClip = m.getVideoPlayer(d);
      m.bringVideoPlayerToFront(d);
      var vp:MovieClip = other;
      TransitionManager.start(vp,{type:mx.transitions.Fade,
         direction:0,
         duration:4,
      easing:mx.transitions.easing.None.easeNone,
         param1:empty,
         param2:empty});

vp.__transitionManager.addEventListener("allTransitionsInDone",transDone);
      }
}

video1Button.onPress = function() {
   buttonTransition(_level0.vidcomp, 1);
};

video2Button.onPress = function() {
   buttonTransition(_level0.vidcomp, 2);
};

video3Button.onPress = function() {
   buttonTransition(_level0.vidcomp, 3);
};
```

Na pierwszy rzut oka jest to naprawdę duża ilość kodu, ale w rzeczywistości jest on oparty na poprzednim ćwiczeniu. Różnica polega głównie na wykorzystaniu klasy `TransitionManager` w celu aktywowania przejścia `Fade` oraz trzech przycisków, które są powiązane z wykorzystywanymi w tym ćwiczeniu filmami.

W kodzie ActionScript z poprzedniego ćwiczenia pojawiły się także dwie inne zmiany. Komenda `eready` czeka, dopóki nie zostaną wczytane wszystkie filmy (sprawdzenie następuje przy pomocy licznika), a następnie sprawia, by każda instancja filmu wideo została odtworzone. Kod funkcji `ecomplete` powoduje po prostu, że każda instancja wideo zostanie odtworzona po zakończeniu odtwarzania poprzedniego filmu.

Zamiast analizować cały przedstawiony powyżej kod ActionScript, przyjrzymy się raczej jego najważniejszym częściom.

Pierwszy wiersz wskazuje programowi Flash, gdzie ma szukać przejść. Jeśli chcesz umieścić go w swoim własnym skrypcie, pamiętaj, by znajdował się w pierwszym wierszu kodu:

```
import mx.transitions.*;
```

Przejścia (`Transitions`) są klasą, a klasy posiadają organizację przypominającą strukturę katalogów. W świetle terminologii związanej z programowaniem obiektowym określeniem, które zastępuje katalog (*directory*), jest pakiet (*package*). Pakiety importuje się za pomocą komendy `import`. W naszym przypadku komenda oznacza: „Importuj przejścia z podpakietu o nazwie `transitions`, który znajduje się w pakiecie `mx`". Wstawienie do kodu gwiazdki oznacza, że chcemy, by program Flash importował wszystkie klasy, które znajdzie w podpakiecie `transitions`.

Pierwsza część kodu to funkcja, jedynym jej zadaniem jest określenie, które z instancji będą widoczne na scenie:

```
function transDone(e) {
   vidcomp.visibleVideoPlayerIndex = e.target.content._name;
   trace(e.target.content._name);
}
```

Funkcja `trace()` umieszcza nazwę widocznego w danej chwili komponentu *FLVPlayback* w panelu *Output* okna *Property inspector*.

Druga funkcja to już przejście właściwe, które aktywuje się, gdy zostanie wciśnięty jeden z trzech znajdujących się na scenie przycisków.

```
49 function buttonTransition(m:MovieClip, d:Number) {
50     if (d != m.visibleVideoPlayerIndex) {
51         var other:MovieClip = m.getVideoPlayer(d);
52         m.bringVideoPlayerToFront(d);
53         var vp:MovieClip = other;
54         //this is a fade transition
55         TransitionManager.start(vp,{type:mx.transitions.Fade,
56                         direction:Transition.IN,
57                         duration:4,
58                         easing:mx.transitions.easing.None.easeNone,
59                         param1:empty,
60                         param2:empty});
61         vp.__transitionManager.addEventListener("allTransitionsInDone",transDone);
62     }
63 }
```

W pierwszym wierszu funkcji nadawana jest nazwa. Nazwany zostaje też komponent *FLVPlayback* (jest to klip filmowy `m`), a także numer instancji (trzy zostały utworzone już wcześniej) — `d`.

```
function buttonTransition(m:MovieClip, d:Number) {
```

Następna część funkcji odnajduje widoczny w danej chwili klip filmowy, a także klip filmowy, który nie jest widoczny, ale powiązany z wciśniętym przyciskiem. Drugi z wymienionych klipów zostanie umieszczony przed tym, który jest w danej chwili odtwarzany: `bringVideoPlayerToFront(d);`.

```
if (d != m.visibleVideoPlayerIndex) {
   var other:MovieClip = m.getVideoPlayer(d);
   m.bringVideoPlayerToFront(d);
   var vp:MovieClip = other;
```

Instancje te zostały zamienione miejscami — ta, która znajduje się z przodu, musi zacząć pojawiać się, wykorzystując w tym celu przejście `Fade`. Efekt ten osiągniemy za pomocą klasy `TransitionManager`.

```
TransitionManager.start(vp,{type:mx.transitions.Fade,
         direction:Transition.IN,
         duration:4,
         easing:mx.transitions.easing.None.easeNone,
         param1:empty,
         param2:empty});
   vp.__transitionManager.addEventListener("allTransitionsInDone",transDone);
   }
}
```

Pierwszy wiersz tego kodu odwołuje się do metody `TransitionManager.Start()`, która tworzy nową instancję `TransitionManager`, określa obiekt docelowy (`vp`), przypisuje przejście (`Fade`) metodą `easing` (`easing:mx.transitions.easing.None.easeNone`) i wykonuje przejście (`vp.__transitionManager.addEventListener("allTransitionsInDone",transDone);`) — wszystko za jednym razem.

Nie możesz oczekiwać, że po prostu wczytasz przejście `Fade` i będzie ono działało. Klasa ta wymaga podania jej pełnej nazwy (`mx.transitions.Fade`) jako parametru klasy `TransitionManager`. Po ustawieniu przejścia `Fade` musisz ustalić wartość kierunku (użyj stałej `Transition.IN`, by osiągnąć efekt pojawiania się) i czas trwania efektu. W tym przypadku materiał wideo będzie pojawiał się w przeciągu pół sekundy.

Metoda `easing` jest zazwyczaj zarezerwowana dla przejść, które przemieszczają klip filmowy po scenie w sposób fizyczny. W tym przypadku możemy określić w czasie trwania przejścia miejsce, w którym zadziała metoda `easing` — na początku, końcu lub w obu miejscach. Dostępne są cztery metody `easing`:

- **`easeIn:`** efekt aktywuje się na początku przejścia,
- **`easeOut:`** efekt aktywuje się na końcu przejścia,
- **`easeInOut:`** efekt aktywuje się zarówno na początku, jak i końcu przejścia,
- **`easeNone:`** brak obliczeń `easing`; wartość ta została użyta w naszym przykładzie.

Ostatnia partia kodu kieruje tym, co dzieje się po wciśnięciu każdego z przycisków.

```
video1Button.onPress = function() {
   buttonTransition(_level0.vidcomp, 1);
};
```

Funkcja `buttonTransition()` jest aktywowana za pomocą wciśnięcia przycisku; komponent *FLV-Playback* jest przenoszony na wierzch stosu głównej listwy czasowej i otrzymuje identyfikator `ID – 1`.

```
49 function buttonTransition(m:MovieClip, d:Number) {
50     if (d != m.visibleVideoPlayerIndex) {
51         var other:MovieClip = m.getVideoPlayer(d);
52         m.bringVideoPlayerToFront(d);
53         var vp:MovieClip = other;
54         //this is a fade transition
55         TransitionManager.start(vp,{type:mx.transitions.Fade,
56                         direction:Transition.IN,
57                         duration:4,
58                         easing:mx.transitions.easing.None.easeNone,
59                         param1:empty,
60                         param2:empty});
61         vp.__transitionManager.addEventListener("allTransitionsInDone",transDone);
62     }
63 }
64
65 video1Button.onPress = function() {
66   buttonTransition(_level0.vidcomp, 1);
67 };
68
69 video2Button.onPress = function() {
70   buttonTransition(_level0.vidcomp, 2);
71 };
72
73 video3Button.onPress = function() {
74   buttonTransition(_level0.vidcomp, 3);
75 };
```

Czego się nauczyłeś?

W czasie tej lekcji:

- Zawarłeś materiał wideo w obiekcie wideo (strony od 279 do 282),
- Utworzyłeś obiekty NetConnection i NetStream w celu przesłania strumieniowego materiału wideo z serwera (strony od 282 do 284),
- Dodałeś elementy kontrolne do obiektu wideo (strony od 285 do 289),
- Wykorzystałeś akcje nasłuchujące w celu przewijania odtwarzania sekwencyjnego (strony od 289 do 296),
- Poznałeś klasę TransitionManager (strony od 296 do 303),
- Dodałeś do filmu przejście typu Fade (strony od 296 do 303).