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

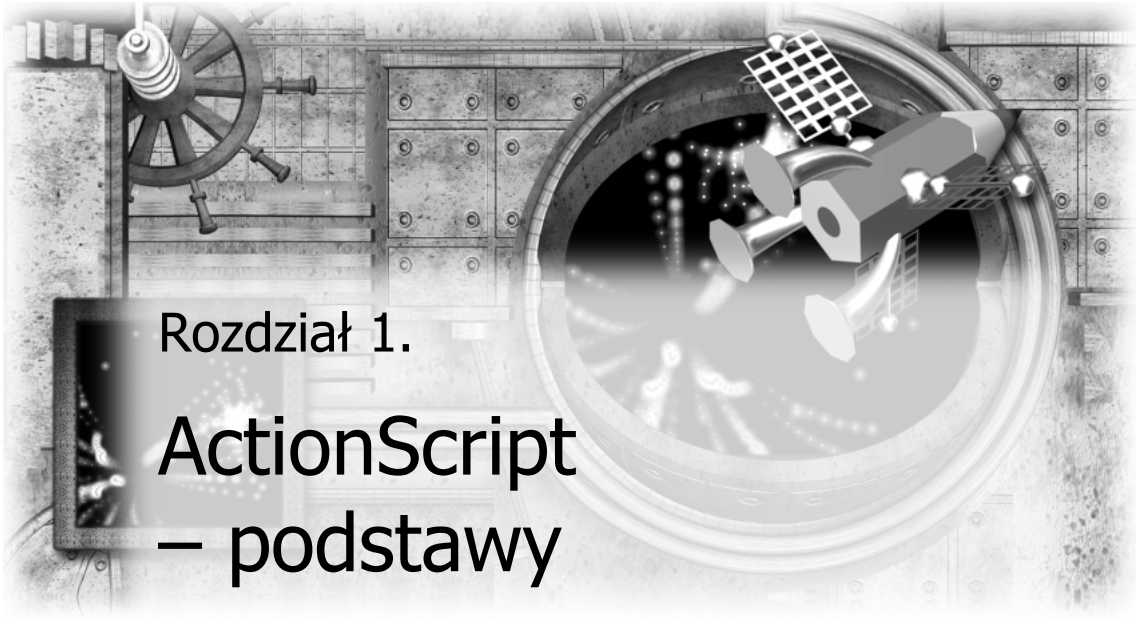
Flash 5 f/x

Autor: Bill Sanders
Tłumaczenie: Wojciech Pazdur
ISBN: 83-7197-289-X
Tytuł oryginału: Flash 5 f/x and Design
Format: B5, stron: 364+8
Zawiera CD-ROM

Książka przedstawia Flasha na poziomie średniozaawansowanym i jest adresowana do dwóch grup użytkowników. Pierwszą z nich stanowią osoby znające Flasha 4 i pragnące szybko „przesiąść sił”na Flasha 5, dlatego właśnie książkę rozpoczyna opis nowości Flasha 5.
Książka przeznaczona jest także dla doświadczonych projektantów i twórców prezentacji, którzy chcieliby dołączyć do arsenału swych umiejętności także korzystanie z Flasha 5. Twórcy stron i prezentacji internetowych, którzy znają się na grafice komputerowej, rysunku, projektowaniu i innych zagadnieniach związanych ze sztuką cyfrową nie potrzebują specjalnego przeszkolenia. Wystarczy im wiedza na temat możliwości Flasha 5 i nabycie umiejętności korzystania z poszczególnych narzędzi programu. Tekst, ilustracje i przykłady zawarte w tej książce mają na celu pokazanie Flasha od jego podstaw, aż do sztuki tworzenia filmów. Przedstawiłem tutaj takie aspekty projektowania jak kompozycja kolorów czy aranżacja obiektów, mając na celu nie tylko zaprezentowanie gotowych rozwiązań, ale nauczenie cię, jak z pomocą Flasha można zrealizować własne projekty i ambicje artystyczne.

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

Spis treści



Rozdział 1.

ActionScript – podstawy

ActionScript to język skryptowy, który pozwala twórcom wzbogacać filmy Flasha. W tym rozdziale omawiamy podstawowe obiekty stosowane w skryptach i wyjaśniamy powiązania poszczególnych elementów Flasha z językiem ActionScript.

Możliwości języka ActionScript

Jeśli pracowałeś już we Flashu, prawdopodobnie używałeś również języka ActionScript. Gdy choć raz umieściłeś polecenie **stop()** w ujęciu kluczowym lub sprawiłeś, by przycisk uruchamiał odtwarzanie za pomocą polecenia **play()**, znaczy to, że posługiwałeś się językiem ActionScript. Język ten jest ściśle powiązany z Flashem i wszystkie jego polecenia odnoszą się do obiektów w filmie Flasha. Na przykład polecenia **play()** i **stop()** pozwalają uruchomić lub zatrzymać odtwarzanie klipu filmowego lub głównego filmu. W innym języku skryptowym czy programistycznym, polecenia te byłyby bardziej ogólne i ich implementacja wymagałaby większej ilości informacji. We Flashu, polecenie **stop()** umieszczone w ujęciu oznacza jedno – zatrzymaj odtwarzanie filmu w tym ujęciu. Język ActionScript jest tak prosty, że możesz nawet nie zdawać sobie sprawy, że go używasz.

Gdy zdobędziesz już pojęcie na temat poleceń języka ActionScript, możesz zastanawiać się, czy można coś zrobić we Flashu bez skryptów. Odpowiedź jest prosta – niewiele (szczególnie wtedy, gdy projektujesz interaktywny film Flasha). Ale nawet duży film Flasha może być bierny. Użytkownicy tradycyjnych stron WWW lubią, gdy coś się dzieje na oglądanej stronie, jednak nie są zaangażowani w akcję; po prostu siedzą przed monitorem zauroczeni tym, co widzą na ekranie, po czym klikają, by przejść w nowe miejsce.

Tworząc skrypty, możesz wyposażyć film Flasha w mechanizm decyzyjny, który zaangażuje użytkownika w akcję. Gdy stworzysz zestaw wyrażeń warunkowych, użytkownik będzie mógł wybrać jedną z opcji i film zareaguje na wybór w odpowiedni sposób. Możesz na przykład stworzyć skrypt, który pozwoli użytkownikowi trzykrotnie udzielić odpowiedzi na pytanie. Gdy za trzecim razem odpowiedź jest błędna, film kieruje użytkownika na „dodatkowy

kurs", gdzie będzie mógł zdobyć dodatkowe informacje przed kolejną próbą. Dodatkowy kurs jest ujęciem oznaczonym etykietą i zawierającym informacje związane z pytaniem. Trzy szanse odpowiedzi to po prostu zapętlone ujęcie (lub skrypt), zliczające liczbę powtórzeń przed wysłaniem użytkownika do ujęcia z dodatkowym kursem.

Gdy na przykład chcesz stworzyć grę zręcznościową, musisz sprawić, by Flash szybko i dokładnie reagował na przeróżne działania użytkownika. Proste, stare filmy Flasha bez skryptów nie reagowały na nic. Dzięki skryptowi Flash może odczytywać położenie kursora, położenie obiektów na ekranie oraz zależności pomiędzy działaniami użytkownika, obiektami, czasem i umiejętnościami gracza. Jednocześnie film może zliczać punktację. Wszystko możesz opanować za pomocą języka ActionScript.

Gdy głównym zastosowaniem Twoich filmów Flasha są strony WWW, skrypty mogą również pełnić rolę pośrednika. Gdy użytkownik wprowadza dane w filmie Flasha, mogą one być przesłane na stronę WWW i użyte na tej stronie lub *vice versa* – możesz przesłać dane formularza ze strony WWW do filmu Flasha. Co istotniejsze – możesz pobrać informacje za pomocą skryptu i przesłać je bezpośrednio do serwera, gdzie mogą być zapisane w bazie danych. Dzięki temu na stronie WWW możesz umieścić interfejs, za pomocą którego użytkownicy będą wprowadzali swoje dane personalne, adresy i numery kart kredytowych, przesyłane następnie do serwera przetwarzającego zamówienia.

Twórcy języków programistycznych i skryptowych są zawsze zadziwieni, gdy programiści uzyskują rezultaty, do których dany język nie był pierwotnie przystosowany. Choć głównym zastosowaniem języka ActionScript jest współpraca z użytkownikiem w filmie Flasha, typ i zasięg interaktywności zależy jedynie od wyobraźni projektanta filmu.

Komfort pracy

W poprzednich wersjach Flasha skrypty tworzyliśmy w małym oknie, w którym poszczególne akcje wybieraliśmy z listy. W oknie mieścił się jedynie fragment skryptu i, choć można było tworzyć naprawdę interesujące skrypty, zadanie to było dosyć żmudne. We Flashu 5 język ActionScript został rozbudowany do pełnowartościowego języka skryptowego, zorientowanego obiektowo. Jest on bardzo podobny do języka JavaScript, jednak tworzenie skryptów jest wspomagane przez interfejs, w którym wskazujemy i klikamy poszczególne akcje. Zaawansowani użytkownicy mogą skorzystać z trybu eksperta (**Expert Mode**), który pozwala wprowadzać kod bezpośrednio z klawiatury tak, jak to się dzieje w zwykłym edytorze tekstu (choć nadal można używać myszy do wskazywania i klikania opcji poszczególnych akcji, funkcji, operatorów i obiektów). Edytor skryptów, zawarty w panelu **Object Actions** (akcje obiektów) lub **Frame Actions** (akcje ujęć) ułatwi Ci naukę tworzenia skryptów, a gdy zdobędziesz wprawę, usunie się z drogi, zapewniając pełną swobodę działania.

Modułowa budowa skryptów

Skrypty we Flashu mają budowę modułową. Oznacza to, że są one niewielkimi modułami, wykonującymi określone operacje, takie jak zatrzymanie czy uruchomienie odtwarzania filmu. Każdy moduł działa samodzielnie, lecz jest powiązany z innymi elementami filmu Flasha. Skrypty możesz przypisywać do przycisków, klipów filmowych i ujęć; gdy film Flasha dojdzie do określonego ujęcia lub klipu filmowego, bądź gdy użytkownik filmu aktywuje przycisk, uruchomi się odpowiedni skrypt.

Nie oznacza to, że skrypty nie mogą być skomplikowane. Mogą. Zwykle jednak wystarczają niewielkie skrypty, umieszczone w przyciskach, klipach filmowych lub ujęciach.

Oto dobra nowina dla osób, które nigdy wcześniej nie programowały: nie musisz uczyć się całego, złożonego języka programowania. Wystarczy, byś opanował elementy, które w danej chwili są Ci potrzebne i użył ich w przyciskach, klipach filmowych i ujęciach filmu. Gdy będziesz chciał, by Twój film miał większe możliwości, po prostu nauczysz się kolejnych elementów języka ActionScript. Doświadczeni programiści stwierdzą, że nauka języka ActionScript ogranicza się do zrozumienia, gdzie i w jaki sposób stosować tradycyjne struktury programistyczne, takie jak wyrażenia warunkowe, tablice czy pętle.

ActionScript zorientowany obiektowo

Miłą niespodzianką dla programistów (i zagadnieniem, które z pewnością powinny zrozumieć osoby nie zajmujące się programowaniem) jest fakt, że język ActionScript jest zorientowany obiektowo. Nie wdając się w aspekty techniczne i terminy naukowe, podejście to koncentruje się na traktowaniu wszystkich elementów programu jako obiektów. Scena, ujęcie, pole tekstowe, rysunek czy symbol – wszystko to są obiekty. Posiadają one swoje atrybuty, które można modyfikować za pomocą skryptów; każdy obiekt ma też swój identyfikator, dzięki któremu możemy się odnosić do obiektu z poziomu skryptu. Na przykład film może zawierać klip filmowy o nazwie **Cecil**. **_rotation** (obrót) to właściwość (atrybut) obiektu o nazwie **Cecil**. W skrypcie możemy umieścić polecenie, które obróci obiekt **Cecil** o 90° (tabela 1.1).

Tabela 1.1. *Właściwości przykładowego klipu filmowego*

Obiekt	Właściwość (atrybut)	Identyfikator ID	Polecenie
Movie Clip (klip filmowy)	_rotation (obrót)	Cecil	„Cecil obróć o 90°”

Obiekty powinny być ustawione kolejno według określonej hierarchii. Każdy z nich może się bowiem składać z kilku innych obiektów. Na przykład klip filmowy może być zbudowany z symboli, rysunków, przycisków, a nawet innych klipów filmowych, przy czym każdy z elementów umieszczony jest na oddzielnej warstwie. Każdy element może z kolei zawierać inne obiekty. Rozważmy następującą sytuację:

- **Zoo** – główna listwa czasowa (główny film);
- **Lion** (lew) – klip filmowy na głównej listwie czasowej;
- **Teeth** (ząb) – klip filmowy, który jest częścią klipu filmowego **Lion;**
- **White** (biały) – właściwość klipu filmowego **Teeth** (kolor biały ma szesnastkowy kod FFFFFF).

Obiekt o nazwie **Zoo** to główna listwa czasowa (najwyższy poziom w hierarchii filmu – **_root**). Obiekt **Zoo** zawiera klip filmowy o nazwie **Lion**. Z kolei **Lion** jest zbudowany (między innymi) z obiektu o nazwie **Teeth**. Jeśli obiekt się zmienia, zmieniają się też obiekty, znajdujące się wyżej w hierarchii. Załóżmy na przykład, że obiekt **Teeth** zmieni kolor (obiekt **Color**) z białego na złoty. Od tej chwili obiekt **Lion** będzie miał złote zęby zamiast białych. To z kolei wpłynie na wygląd całego obiektu o nazwie **Zoo**. Pierwotnie **Zoo** zawierało lwa o białych zębach – teraz lew posiada złote wstawki. **Zoo** nie wygląda już tak samo.

Hierarchia ułożona dla tego przykładu zoologicznego, w skrypcie wyglądałaby następująco:

```
toothColor=new Color(_root.lion.teeth);
toothColor.setRGB(0xFF9900);
```

Zmiany koloru zęba może dokonać skrypt przycisku, ujęcia lub innego klipu filmowego; może to być również skrypt zawarty w samym klipie filmowym **Teeth**. Jeśli tylko obiekt jest odpowiedniego rodzaju (klip filmowy), dowolny obiekt może zawierać skrypt, kierujący do niego polecenia. Jednak skrypt musi zawierać odpowiednią ścieżkę adresową, sporządzoną zgodnie z zasadami składni. W powyższym przykładzie ścieżka ma postać:

```
_root.lion.teeth;
```

Skrypt zawarty w dowolnej hierarchii może przesyłać polecenia do dowolnie wybranej hierarchii. Skrypt dowolnego obiektu może wpływać na zachowania i atrybuty tego lub innych obiektów. Na przykład obiekt, który jest „potomkiem" w hierarchii klipu filmowego, może zawierać skrypt odnoszący się do „prapotomka" (czyli „potomka potomka") innego klipu filmowego – czyli innej hierarchii – i zlecić mu przejście do ujęcia 34. Równie dobrze może mu zlecić podwojenie wymiarów. Z kolei ujęcie numer 34 może zawierać dodatkowy skrypt, zlecający obiektowi jeszcze innego klipu filmowego, by zagrał określony dźwięk.

Praca ze skryptami i obiektami nie jest tak skomplikowana, jak wygląda to na pierwszy rzut oka. Dzięki modułowej budowie skryptów we Flashu, nie musisz tworzyć jednego ogromnego programu o długości 29876 linii, gdyż film zawiera krótkie skrypty umieszczone w ujęciach, przyciskach i klipach filmowych. Gdy tworzysz film Flasha (czy to z użyciem skryptów, czy też bez nich), pamiętaj, by uporządkować poszczególne jego części i zaplanować użycie poszczególnych ujęć kluczowych. Praca w środowisku zorientowanym obiektowo również wymaga planowania; pomaga to efektywnie sterować filmem za pomocą skryptów. Język ActionScript możesz uważać za prosty, lecz efektywny język programowania zorientowany obiektowo, który znacznie ułatwia pracę.

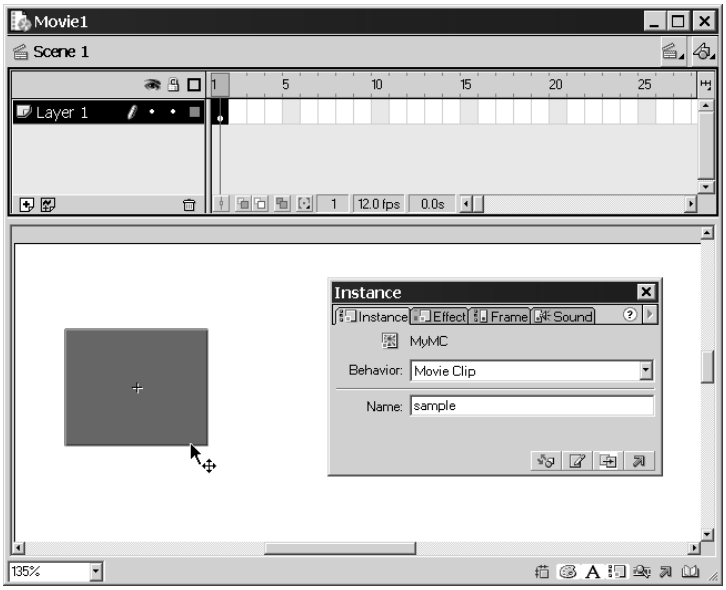
Gdzie umieszczamy skrypty?

Mówiąc najprościej – skrypty dołączamy do klonów klipów filmowych, przycisków i ujęć. Jednakże stan klipu filmowego, przycisku lub ujęcia może być zmieniany przez wiele różnych czynników. Zanim rozpoczniesz tworzenie skryptów, trzeba wyjaśnić, w jakich okolicznościach możesz umieszczać je w klipach filmowych, przyciskach i ujęciach filmu Flasha.

Klipy filmowe

We Flashu 5 klipy filmowe mogą zawierać własne skrypty. W poprzednich wersjach programu, skrypty można było umieszczać jedynie w przyciskach i ujęciach. Klipy filmowe są jedynymi obiektami, których właściwości można zmieniać za pomocą skryptów, więc oferują one największe możliwości w dziedzinie współpracy ze skryptami Flasha 5. Inne obiekty można modyfikować poprzez wykonywanie skoków do miejsc listwy czasowej, w których obiekty ulegają zmianie w związku z automatyczną animacją lub nową konfiguracją w ujęciu kluczowym. Jedynie klipy filmowe posiadają właściwości, które można adresować i zmieniać bezpośrednio z poziomu skryptu. Aby stworzyć klip filmowy, wykonaj następujące czynności:

1. Wybierz polecenie **File/New** (klawisze **Ctrl+N**), by utworzyć nowy dokument Flasha.
2. Wybierz narzędzie **Rectangle** Tool (prostokąt) w przyborniku (pasek **Tools**). Narysuj na obrazie prostokąt i zaznacz go za pomocą narzędzia **Arrow** Tool (strzałka). (Wybierz narzędzie **Arrow** Tool w przyborniku; kliknij myszą i przeciągnij na obrazie, by otoczyć cały narysowany prostokąt ramką zaznaczenia).
3. Wybierz polecenie **Insert/Convert to Symbol** lub naciśnij klawisz **F8**. W okienku **Symbol Properties**, które się pojawi, w polu **Behavior** (zachowanie) wybierz opcję **Movie Clip** (klip filmowy). W polu **Name** wprowadź nazwę **MyMC**. Kliknij **OK**.
4. Wybierz polecenie **Window/Panels/Instance** lub naciśnij klawisze **Ctrl+I**, by wyświetlić panel **Instance**. W polu **Name** tego panelu wprowadź nazwę klonu **sample** (rysunek 1.1).



Rysunek 1.1. *W panelu Instance wprowadź nazwę klonu klipu filmowego*

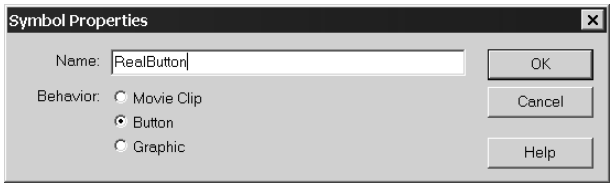
Utworzony klip filmowy jest gotów do działania. Na rysunku 1.1 pokazaliśmy nowy klip filmowy oraz panel **Instance**, który jest niezbędny do pracy z klonami klipów filmowych (i innych symboli). Nazwa klonu klipu filmowego jest w tym momencie ważniejsza od nazwy samego symbolu, widocznej nad rozwijaną listą **Behavior**.

Przyciski

Zanim stworzono Flasha 5, przyciski były jedynym typem symboli, w których można było umieszczać skrypty. Choć teraz można je umieszczać również w klipach filmowych, przyciski nadal pełnią kluczową rolę w filmach Flasha sterowanych skryptami. Każdy użytkownik Flasha wie, że przycisk – symbol o zachowaniu **Button** – wcale nie musi wyglądać jak przycisk. Przycisk może być częścią większego elementu graficznego, przygotowanego na działanie użytkownika. Na przykład symbol **Button** może mieć postać osoby, która siedzi z innymi osobami przy stole konferencyjnym. Gdy klikniesz ten przycisk-osobę, film przeskoczy do ujęcia, w którym pojawi się informacja na temat funkcji danej osoby.

Aby stworzyć symbol przycisku, wykonaj następujące czynności:

1. Wybierz polecenie **File/New** (klawisze **Ctrl+N**), by utworzyć nowy dokument Flasha.
2. Kliknij dwukrotnie nazwę pierwszej warstwy (**Layer 1**) po prawej stronie listwy czasowej. Gdy to zrobisz, będziesz mógł wprowadzić nową nazwę warstwy. Nadaj warstwie nazwę **Real-Button**.
3. Wybierz narzędzie **Oval** Tool (owal) w przyborniku i narysuj na obrazie koło.
4. Kliknij warstwę **Real-Button** po prawej stronie listwy czasowej. Program zaznaczy narysowane przed chwilą koło.
5. Wybierz polecenie **Insert/Convert to Symbol** lub naciśnij klawisz **F8**.
6. W okienku **Symbol Properties**, które się pojawi, w polu **Behavior** wybierz opcję **Button** (przycisk). W polu **Name** wprowadź nazwę **RealButton** (rysunek 1.2). (Gdy wybierzesz zachowanie **Button** w polu **Behaviour**, nowy symbol automatycznie uzyska cztery ujęcia – **Up**, **Over**, **Down** i **Hit**. Ujęcia te można edytować w trybie edycji symboli. Nie są one istotne dla samego skryptu, lecz mają wpływ na współpracę przycisku z użytkownikiem. Za pomocą przycisku użytkownik będzie uruchamiał zawarte w nim skrypty).

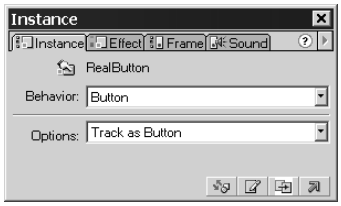


Rysunek 1.2. *W oknie Symbol Properties definiujemy symbol przycisku*

Możesz też użyć jednego z gotowych przycisków, dostarczonych wraz z Flashem. Wybierz polecenie **Window/Common Libraries/Buttons**. Pojawi się okno biblioteki **Library – Buttons.fla**. Znajdziesz tam gotowe przyciski. Wybierz jeden z nich, by pojawił się w oknie podglądu biblioteki. Przeciągnij go na obraz, by umieścić jego klon w filmie. Jeśli zamierzasz użyć kilku klonów symbolu, umieść każdy z nich na oddzielnej warstwie. (Flash radzi sobie z kilkoma klonami na pojedynczej warstwie, lecz zarządzanie nimi może być problematyczne).

Właściwości klonów

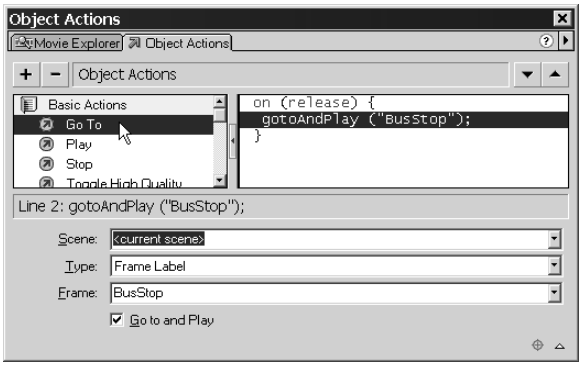
Symbole to elementy zawarte w bibliotece. Gdy pragniemy użyć symbolu w filmie, umieszczamy w nim *klon* symbolu. Klon to odnośnik do symbolu, który znajduje się w bibliotece. Przyciski zawarte w filmie są więc klonami symboli (zawartych w bibliotece) o zachowaniu (**Behavior**) zdefiniowanym jako **Button**. Przyciski posiadają kilka właściwości, które są bardzo ważne dla skryptów. Aby wyświetlić właściwości klonu, zaznacz go na obrazie, a następnie wybierz polecenie **Modify/Instance**. Możesz też wybrać polecenie **Window/Panels/Instance** lub nacisnąć klawisze **Ctrl+I**. Wszystkie te operacje prowadzą do wyświetlenia panelu **Instance**. Na rysunku 1.3 widać panel **Instance** z właściwościami typowego przycisku. Zwróć uwagę na różnice pomiędzy rysunkami 1.1 i 1.3.



Rysunek 1.3. *Panel Instance po zaznaczeniu klonu przycisku*

Na rysunku 1.3 możesz zauważyć, że w liście **Behavior** jest ustawiona opcja **Button**, zaś w liście **Options** – opcja **Track as Button**. Elementy te są wyjątkowo ważne dla skryptu. Opcja **Button** w liście **Behavior** jest informacją, że klon symbolu działa jako przycisk. Jeśli pierwotny symbol jest przyciskiem, program automatycznie wybierze zachowanie **Button** w jego klonie. Gdyby natomiast pierwotny symbol był klipem filmowym lub elementem graficznym, program ustawiłby zachowanie klonu odpowiednio na opcji **Movie Clip** lub **Graphic**. Jednakże zachowanie klonów (lista **Behavior** w panelu **Instance**) można zmieniać niezależnie od zachowania pierwotnych symboli (pole **Behavior** w oknie **Symbol Properties**).

Jednym z ważniejszych elementów panelu **Instance** jest ikona **Edit Actions**, znajdująca się w prawym dolnym narożniku (symbol strzałki). Gdy klikniesz tę ikonę, pojawi się edytor skryptów, zawierający skrypt związany z zaznaczonym klonem przycisku. Rysunek 1.4 przedstawia edytor skryptów z typowym (bardzo prostym) skryptem przycisku.



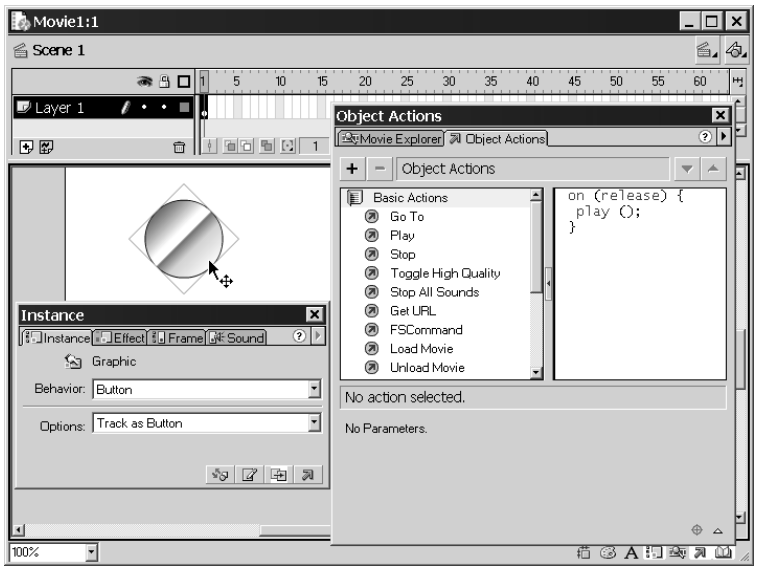
Rysunek 1.4. *Skrypt związany z klonem przycisku*

Oto krótkie ćwiczenie, które przekona Cię, jak ważne są właściwości przycisków:

1. Stwórz przycisk, stosując opisaną wcześniej metodę. Nadaj mu nazwę **RealButton**.
2. Otwórz panel **Instance**.
3. Z rozwijanej listy **Behavior** (zachowanie) wybierz opcję **Graphic** (symbol graficzny). Tym samym zmienisz zachowanie przycisku na zachowanie klonu symbolu graficznego. Zaznacz ten klon na obrazie.
4. Kliknij ikonę **Edit Actions** w panelu **Instance**, by wyświetlić okno **Object Actions** z edytorem skryptów. W edytorze skryptu wszystkie akcje są niedostępne (wyszarzone). Żadnej z nich nie możesz dodać do skryptu.

Choć ikona w oknie biblioteki nadal wskazuje, że nasz symbol jest przyciskiem, jego klon został zmieniony w ten sposób, że stał się klonem symbolu graficznego. Symbolom graficznym nie można przypisywać skrytpów. Innymi słowy – nie możesz użyć symbolu graficznego, by stworzyć skrypt. W oknie **Instance** klonu musi być ustawione zachowanie **Movie Clip** (klip filmowy) lub **Button** (przycisk), byś mógł przypisać do niego choćby pojedynczą akcję.

Pokazaliśmy, że klon przycisku można przekształcić w klon symbolu innego typu; oczywiście, możliwe jest też odwrotne działanie. Możesz stworzyć symbol graficzny lub klip filmowy i zmienić właściwość **Behavior** jego klonu na **Button**, by stał się klonem przycisku. Na rysunku 1.5 pokazaliśmy klon symbolu o nazwie **Graphic**. Pierwotny symbol **Graphic** jest symbolem graficznym, jednak zachowanie jego klonu zmieniono na **Button**. Klon symbolu stał się rzeczywistym przyciskiem, dlatego wszystkie akcje w panelu **Object Actions** są aktywne, podobnie jak skrypt przycisku. W taki sam sposób – w panelu **Instance** – można przekształcać klony klipów filmowych w klony przycisków.



Rysunek 1.5. *Klon symbolu graficznego zmieniony w klon przycisku*

Zmiany właściwości klonów w filmie

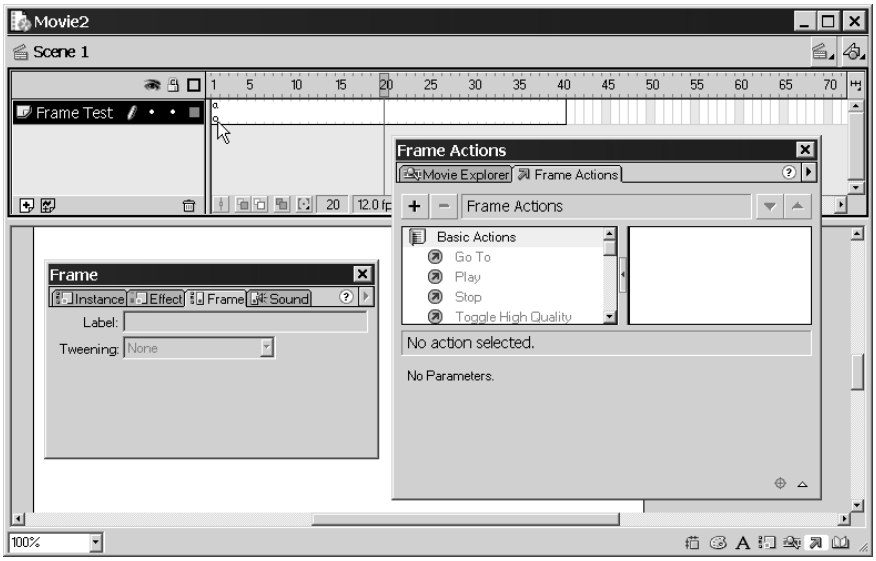
W filmie Flasha obiekty mogą się zmieniać. Dla obeznanych użytkowników Flasha nie jest to żadną nowością; jednakże klony symboli mogą zmieniać swoje zachowanie z **Graphic** na **Button** lub **Movie Clip** w czasie odtwarzania filmu. Na przykład klon symbolu graficznego, który znajduje się w klatce numer 1, może się przekształcić w klon przycisku w klatce numer 10, a następnie w klon klipu filmowego w klatce numer 40. Tego typu zmiany nie są jednak zalecane. Zwracam na to uwagę, ponieważ czasem wprowadzamy je przez pomyłkę, a to może prowadzić do niepożądanych rezultatów.

Właściwości ujęć

Trzeci typ obiektów, w których można umieszczać skrypty, stanowią ujęcia (**Frames**). W przypadku ujęć kluczowych, pustych ujęć kluczowych i zwykłych ujęć pośrednich panel **Frame Actions** udostępnia aktywne akcje.

Jednakże skrypty są wykonywane jedynie w ujęciach kluczowych (również pustych). Gdy umieścisz skrypt w ujęciu pośrednim, zostanie on „przejęty" przez ujęcie kluczowe. By się o tym przekonać, wykonaj poniższy eksperyment:

1. Wybierz polecenie **File/New** lub użyj klawiszy **Ctrl+N**, by utworzyć nowy dokument Flasha. Pojawi się pusty dokument z pojedynczą warstwą o nazwie **Layer 1**. Pierwsze ujęcie warstwy jest ujęciem kluczowym.
2. Zmień nazwę warstwy na **FrameTest**. Zawsze warto nadawać własne nazwy wszystkim elementom filmu.
3. Na listwie czasowej kliknij klatkę numer 40, by ją zaznaczyć. Program podświetli zaznaczoną klatkę.
4. Wybierz polecenie **Insert/Frame** lub naciśnij klawisz **F5**. Program stworzy puste ujęcia pośrednie we wszystkich klatkach (aż do klatki numer 40).
5. Zaznacz ujęcie w klatce numer 20 i wybierz polecenie **Modify/Frame** (możesz również użyć klawiszy **Ctrl+F**). Pojawi się panel **Frame**.
6. Kliknij ikonę **Show Actions** (symbol strzałki) w prawym dolnym narożniku okna dokumentu, by wyświetlić panel **Frame Actions**. W prawym oknie edytora skryptów zaznacz kategorię **Basic Actions**. Po rozwinięciu listy, kliknij dwukrotnie akcję **stop()**, by umieścić ją w skrypcie. Na rysunku 1.6 pokazaliśmy, jak powinna wyglądać w tym momencie listwa czasowa.



Rysunek 1.6. *W panelu Frame Actions edytujemy skrypty ujęć*

Na rysunku 1.6 zwróć uwagę na małą literę ***a*** na listwie czasowej, która występuje w początkowym ujęciu kluczowym. Oznacza ona, że ujęcie to zawiera skrypt. Jednakże skrypt wprowadziliśmy w ujęciu numer 20. Aby dokończyć eksperyment, wykonaj jeszcze trzy polecenia:

1. Zmień ujęcie w klatce numer 10 w ujęcie kluczowe (kliknij klatkę numer 10 i naciśnij klawisz **F6** lub wybierz polecenie **Insert/Keyframe**).
2. Zaznacz ujęcie w klatce numer 20 i wybierz polecenie **Modify/Frame**. Pojawi się panel **Frame**.
3. Kliknij ikonę **Show Actions** (symbol strzałki) w prawym dolnym narożniku okna dokumentu, by wyświetlić panel **Frame Actions**. Edytor skryptów nie wyświetli już żadnego skryptu. Ten, który wprowadziliśmy wcześniej, znikł.

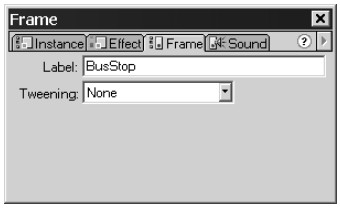
Zniknięcie skryptu z ujęcia pośredniego w powyższym ćwiczeniu jest całkowicie prawidłowe. Warto wyjaśnić, dlaczego tak się stało. Gdy wprowadzamy skrypt w ujęciu pośrednim, jest on „przejmowany" przez poprzednie ujęcie kluczowe w warstwie. Gdy bezpośrednio po wprowadzeniu skryptu sprawdzisz inne ujęcia pośrednie związane z tym samym ujęciem kluczowym, zobaczysz ten sam skrypt. Gdy zmodyfikujesz skrypt w ujęciu numer 5, zmieni się on również w ujęciu kluczowym poprzedzającym ujęcie numer 5. Zmodyfikowany skrypt pojawi się także we wszystkich ujęciach pośrednich, związanych z tym ujęciem kluczowym. W rzeczywistości skrypt jest zawarty tylko w ujęciu kluczowym, zaś ujęcia pośrednie po prostu wyświetlają zawartość poprzedzającego je ujęcia kluczowego.

Podkreślmy – skrypty mogą być przechowywane i wykonywane wyłącznie w ujęciach kluczowych. Ujęcia pośrednie nie zawierają własnych skryptów, więc wyświetlają jedynie skrypty zawarte w poprzedzającym je ujęciu kluczowym , ale ich nie wykonują. W powyższym eksperymencie pozornie wprowadziliśmy akcję **stop()** w ujęciu pośrednim, lecz w rzeczywistości stała się ona akcją ujęcia kluczowego, poprzedzającego to ujęcie pośrednie. Skrypty możemy więc wprowadzać – dla wygody – w dowolnych ujęciach, lecz są one przechowywane i wykonywane wyłącznie w ujęciach kluczowych.

Etykiety ujęć

Właściwości ujęć są stosunkowo proste w porównaniu z przyciskami. W polu **Label** panelu **Frame**, ujęcie możesz opatrzyć *etykietą* lub *komentarzem*.

Aby opatrzyć ujęcie etykietą, zaznacz je na listwie czasowej i wprowadź nazwę etykiety w polu **Label** panelu **Frame**. Aby opatrzyć ujęcie komentarzem, wykonaj te same czynności, lecz przed treścią komentarza umieść dwa ukośniki (//). Korzystając z etykiet możesz w wygodny sposób odnosić się do ujęć w skryptach. Gdy tworzysz skrypt, często wykonywaną operacją jest skok do określonego ujęcia i wykonanie tam jakiegoś zadania. Poprawne adresowanie ujęć jest w tym przypadku niezwykle ważne. Gdy opatrzysz ujęcie opisową etykietą, będziesz mógł się do niego odnosić z poziomu skryptu w prosty i przejrzysty sposób. Na rysunku 1.7 widać na przykład etykietę **BusStop**. Jej nazwa od razu sugeruje, że dane ujęcie przedstawia zatrzymujący się autobus. Tworząc skrypt, możesz zlecić Flashowi skok do ujęcia z etykietą **BusStop** zamiast do ujęcia w klatce numer 83 (a czy nie była to klatka numer 82? a może 84? chyba nie 48?).

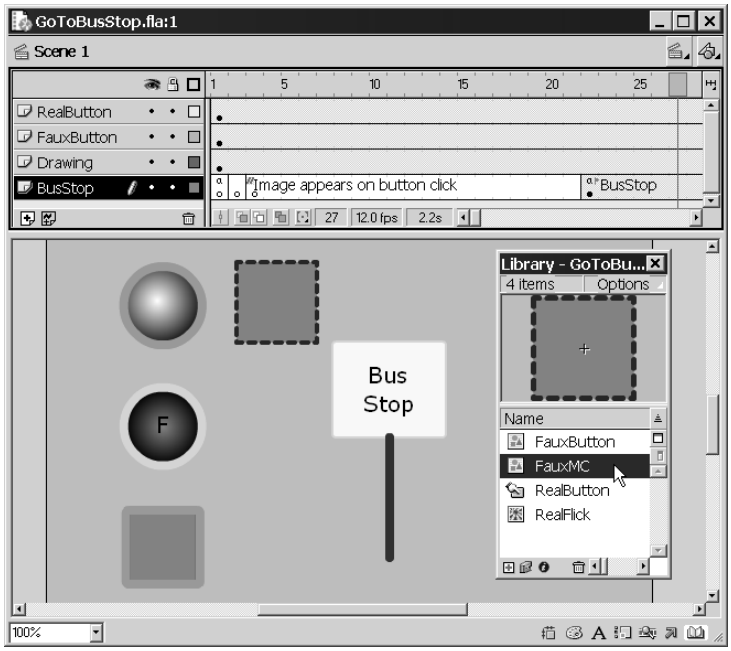


Rysunek 1.7. *Wpisz nazwę etykiety ujęcia w polu Label*

Jeśli jesteś wprawnym programistą, z pewnością starasz się unikać ciągłych skoków w programach. Jednakże zestaw skryptów, które sterują odtwarzaniem filmu, wykonuje skoki na listwie czasowej, a nie w programie. Skrypty dotyczą bowiem obiektów i klatek listwy czasowej. Gdy musisz zrezygnować z sekwencyjnego odtwarzania ujęć i skoczyć do określonego ujęcia w listwie czasowej, opatrz je opisową etykietą, byś mógł odnieść się do niej w akcji **gotoAndPlay**. Duże znaczenie przy tworzeniu skryptów mają również opisowe nazwy obiektów.

Komentarze ujęć

Komentarze pomagają zapamiętać, co się dzieje w określonych punktach filmu. Nie mają one wpływu na jego przebieg. Podobnie jak etykiety, komentarze wprowadzamy w panelu **Frame** w polu **Label**. Jedyna różnica polega na tym, że przed komentarzem wstawiamy dwa ukośniki (//). Rysunek 1.8 przedstawia listwę czasową, w której jedno ujęcie warstwy **BusStop** jest opatrzone etykietą, a drugie komentarzem. Etykiety są oznaczane w listwie czasowej chorągiewką, obok której – jeśli pozwala na to miejsce – znajduje się nazwa etykiety; komentarze są oznaczane symbolem podwójnego ukośnika (//), obok którego – jeśli pozwala na to miejsce – znajduje się treść komentarza.



Rysunek 1.8. *Listwa czasowa wyświetla etykiety i komentarze*

Jak tworzymy skrypty?

Dla osób, które nie zajmują się programowaniem, tworzenie skryptów w języku ActionScript jest niesłychanie proste. Z kolei programiści przy pierwszym kontakcie z tym językiem mogą odczuć cos w rodzaju zaskoczenia, gdyż są oni przyzwyczajeni do potężnych plików tekstowych, zawierających skrypty lub kody źródłowe programów. Skrypty w języku ActionScript tworzymy zaś przez klikanie i dodawanie pojedynczych akcji i niewielkich elementów kodu.

Panele Object Action i Frame Action

Program Flash wyświetla panele **Object Action** i **Frame Action** zamiennie, w zależności od tego, czy zaznaczysz obiekt na obrazie, czy też ujęcie na listwie czasowej. Panele te zawierają edytor skryptu, gdzie możesz wprowadzać i edytować kod. Panel **Object Action** lub **Frame Action** możesz wyświetlić przez wybranie polecenia **Window/Action** lub użycie skrótu klawiszowego **Ctrl+Alt+A**. Możesz też kliknąć ikonę **Show Actions** w prawym dolnym narożniku okna dokumentu lub ikonę **Edit Actions** w prawym dolnym narożniku panelu **Instance**. Abyś mógł tworzyć skrypty i wprowadzać akcje, musi być zaznaczony klip filmowy, przycisk lub ujęcie filmu. Istnieją różne metody wprowadzania akcji i innych elementów skryptów; użyj tej, która najbardziej Ci odpowiada.

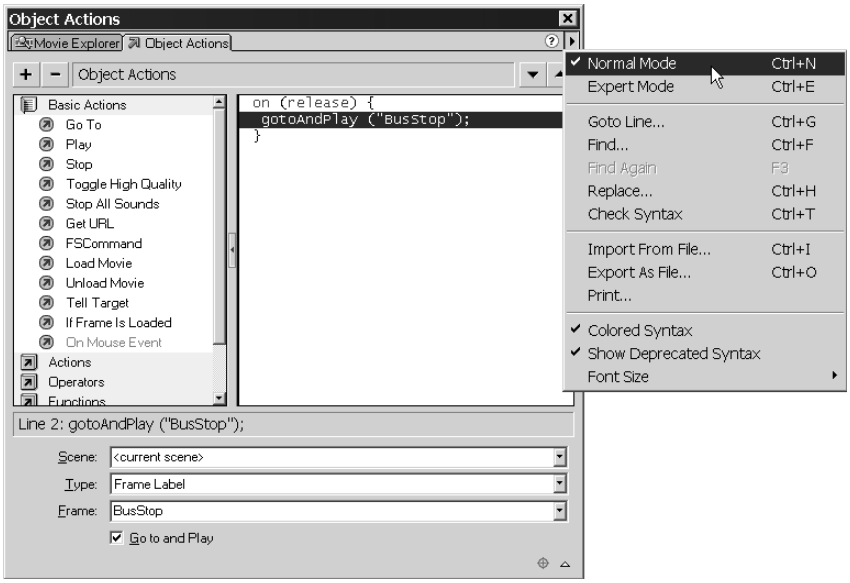
- Przeciągnij i upuść – zaznacz akcję na liście, a następnie przeciągnij ją do okna edytora skryptu.
- Kliknij dwukrotnie – dwukrotne kliknięcie akcji na liście dodaje ją do skryptu.
- Rozwiń menu – jeśli przyzwyczaiłeś się do metody wprowadzania akcji, stosowanej we Flashu 4, kliknij ikonę **+** (plus) w lewym górnym narożniku panelu **Object Actions** lub **Frame Actions** i z rozwiniętego menu wybierz akcję, by dodać ją do skryptu.
- Wprowadź z klawiatury – jeśli jesteś programistą lub po prostu wolisz posługiwać się klawiaturą, wybierz tryb **Expert Mode**. Pozwala on wprowadzać kod skryptu bezpośrednio z klawiatury, czyli bez pomocy (lub ingerencji) programu.
- Łącz metody – w trybie **Expert Mode** możesz korzystać ze wszystkich wymienionych metod. Pod względem edycji skryptów, Flash 5 jest najbardziej elastyczny z dotychczasowych wersji programu.

Tryb Normal Mode

Tryb **Normal Mode** jest zalecany szczególnie dla początkujących użytkowników. Gdy tworzysz skrypt w tym trybie, program wyświetla wszystkie parametry każdego elementu skryptu, niezależnie od tego, czy jest to akcja, funkcja, operator, właściwość czy obiekt. Na rysunku 1.9 pokazaliśmy listę akcji kategorii **Basic Actions**. Skrypt, który jest widoczny w oknie edytora po prawej stronie, zawiera akcję **gotoAndPlay()**. Dolna część panelu – obszar parametrów – wyświetla wszystkie parametry zaznaczonej akcji. Niektóre elementy w obszarze parametrów współdziałają ze sobą. Na przykład, gdy w przypadku akcji **gotoAndPlay()** w polu **Type** wybierzesz opcję **Frame Label** (adresowanie ujęcia za pomocą etykiety), program umieści parametr **Frame** w cudzysłowie, przyjmując, że jest to nazwa etykiety ujęcia. Na rysunku 1.9 widać również rozwijane menu palety **Object Actions**, w którym można wybrać tryb **Normal Mode** lub **Expert Mode**.

W trybie **Normal Mode** okno akcji (po lewej stronie panelu) zawiera dodatkową kategorię **Basic Actions**, która zawiera ponad połowę akcji Flasha 4. Jeśli korzystasz wyłącznie z tych akcji, możesz bez przeszkód wyeksportować film w formacie Flasha 4. Dla ułatwienia nazwy akcji w tej kategorii są wyświetlane jako nazwy angielskie (a nie jako nazwy pojawiające się w kodzie skryptu). Do pozostałych kategorii akcji należy między innymi kategoria **Actions**, zawierająca znacznie większy zestaw akcji; nowe akcje, funkcje, właściwości i obiekty są wyświetlane w formacie Flasha 5, czyli jako nazwy używane w skrypcie.

Gdy w trybie **Normal Mode** tworzysz skrypt dla przycisku, program automatycznie dołącza detektory zdarzeń przycisku, jeśli jest to konieczne; gdy tworzysz skrypt dla klipu filmowego, program automatycznie dodaje detektory zdarzeń klipu.



Rysunek 1.9. *W trybie Normal Mode edytor skryptów służy pomocą podczas wprowadzania akcji i parametrów*

W przykładzie przedstawionym na rysunku 1.9 wybraliśmy akcję **Go To** z kategorii **Basic Actions**, zaś program automatycznie dodał do skryptu detektor zwolnienia klawisza myszy **on (release)**. W przypadku dłuższych skryptów funkcja ta może być uciążliwa, ponieważ program będzie dodawał detektor **on()** lub **onClipEvent()** za każdym razem, gdy umieścisz w skrypcie nową akcję. Niekiedy chcemy w kodzie pojedynczego detektora umieścić kilka akcji lub funkcji – wówczas usuwanie segmentów **on (release)** może być niewygodne.[1]

Tryb Expert Mode

Wybierając tryb **Expert Mode**, rezygnujesz z pomocy programu, dostępnej w trybie **Normal Mode**. Wybierz więc tryb **Expert Mode**, jeśli masz wrażenie, że pomoc programu tylko przeszkadza Ci w pracy. Program nadal będzie udzielał podpowiedzi – w znacznie ograniczonym zakresie – dotyczących elementów, których będą wymagały wprowadzane akcje. Na przykład, gdy dodasz do skryptu akcję **gotoAndPlay()** w trybie **Expert Mode** w skrypcie pojawi się następujący zapis:

```
gotoAndPlay (frame);
```

Program podświetli słowo **frame** wskazując w ten sposób, że powinieneś wprowadzić informację o ujęciu. W trybie **Expert Mode** nie działają jednak funkcje automatycznego umieszczania etykiet w cudzysłowach i inne funkcje wspomagające wprowadzanie parametrów. Na przykład, jeśli zamierzasz przeskoczyć do ujęcia w innej scenie, musisz ręcznie wprowadzić adres sceny oraz ujęcia:

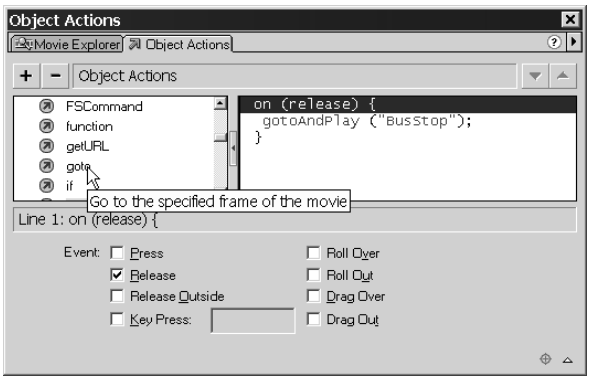
```
gotoAndPlay ("Scene 5", "hit");
```

[1] Możesz jednak uniknąć tego kłopotu. Wystarczy, byś zaznaczył linię wewnątrz kodu detektora, pod którą chcesz umieścić nową akcję. Następnie kliknij dwukrotnie nową akcję, by dodać ją do skryptu. Program „zauważy", że pracujesz wewnątrz kodu detektora i nie doda nowego detektora – *przyp. tłum.*

Program nie dodaje też automatycznie detektorów zdarzeń myszy i klipów filmowych – musisz wprowadzać je samodzielnie. Gdy zdobędziesz wprawę w tworzeniu skryptów, parametry akcji staną się dla Ciebie oczywiste. Jeśli jednak masz z tym kłopoty, oszczędzisz sobie czasu i frustracji, korzystając z trybu **Normal Mode**.

Detektory zdarzeń

Z rozdziale 6. dokładnie omawiamy poszczególne opcje związane z detekcją zdarzeń myszy. W tym miejscu przyjrzyjmy się niektórym opcjom, które niemal zawsze są potrzebne w trakcie tworzenia skryptów dla przycisków. Oprócz zdarzenia **Release** (zwolnienie lewego klawisza myszy), masz do dyspozycji sześć innych zdarzeń, związanych z manipulowaniem myszą. Oprócz tego możesz wykrywać zdarzenie **Key Press**, polegające na wciśnięciu określonego klawisza na klawiaturze. Jeden detektor może wykrywać kilka zdarzeń – wówczas każde z nich uruchamia fragment skryptu, zawarty wewnątrz detektora. W większości zastosowań wykrywamy zdarzenie **Release** – skrypt jest uruchamiany po kliknięciu przycisku i zwolnieniu klawisza myszy. Na przedstawionym na rysunku 1.10 polu **Event** widać poszczególne zdarzenia myszy, wykrywane przez detektor **on**.



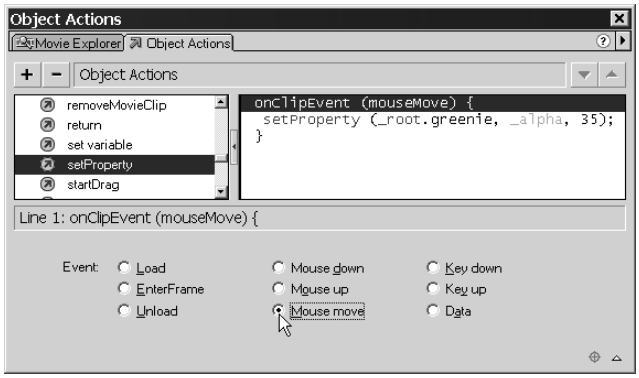
Rysunek 1.10. *W polu Event możesz zaznaczyć zdarzenia, które będą wykrywane przez detektor on*

Klipy filmowe, podobnie jak przyciski, mają swoje detektory zdarzeń. Domyślnym zdarzeniem, które jest wykrywane przez detektor klipu filmowego jest **Load** – zdarzenie polegające na wczytaniu klipu filmowego do pamięci komputera. Na przedstawionym na rysunku 1.11 polu **Event** widać wszystkie zdarzenia klipu filmowego, wykrywane przez detektor **onClipEvent()**.

Zdarzenia **MouseDown** i **MouseUp** detektora **onClipEvent()** odpowiadają zdarzeniom **Press** i **Release** detektora **on()**. Zanim oswoisz się z tym różnicami, pracuj w trybie **Normal Mode**. Możesz również przygotować najtrudniejsze fragmenty skryptu w trybie **Normal Mode**, a następnie przejść do trybu **Expert Mode**, by swobodnie wprowadzać wszelkie zmiany.

Edycja wyrażeń

Flash 4 zawierał edytor wyrażeń (**Expression Editor**), którego funkcje we Flashu 5 przejęły panele **Object Actions** i **Frame Actions**. Większość operacji związanych z edycją wyrażeń wykonujemy podczas dodawania akcji i ich parametrów do skryptu. Wyrażenia możesz edytować w edytorze skryptu zarówno w trybie **Normal Mode**, jak i **Expert Mode**. W trybie **Normal Mode** edycja wyrażeń odbywa się w dolnej części panelu, gdzie ustawiasz parametry akcji, właściwości, funkcji, operatorów i obiektów.



Rysunek 1.11. *Zdarzenia wykrywane przez detektor onClipEvent*

Gdy zaznaczysz element skryptu, w dolnej części panelu pojawią się jego wszystkie parametry, dzięki czemu możesz poprawnie je stosować. W rozdziale 2. omówimy poszczególne typy danych, stosowane w języku ActionScript; wyrażenia w dużej mierze zależą od typów danych.

Klipy filmowe i ścieżki

Klipy filmowe (**Movie Clip**) to jedyne symbole, których właściwości można odczytywać i modyfikować za pomocą skryptów. Choć przyciski (**Button**) również mogą zawierać skrypty, nie można ich modyfikować bezpośrednio za pomocą skrytpów. Symbole graficzne (**Graphic**) nie mogą zawierać skryptów, nie można też odczytywać ani modyfikować ich właściwości za pomocą skryptów. Ujęcia mogą zawierać skrypty, jednak nie posiadają one właściwości, które można by modyfikować (z wyjątkiem zmiany aktualnego ujęcia oraz skoku do ujęcia w klipie filmowym).

Niezależne listwy czasowe

Klipy filmowe posiadają własne listwy czasowe, niezależne od listwy czasowej głównego filmu. Mogą one być odtwarzane nawet wtedy, gdy główny film jest zatrzymany. Jako obiekty klipy filmowe są małymi filmami Flasha, rządzącymi się własnymi prawami. Każdy klip filmowy może przesyłać polecenia do innych klipów filmowych, oddziałując na nie; może też przyjmować polecenia od innych klipów filmowych; może też być sterowany przez główną listwę czasową oraz przyciski i ujęcia zawierające skrypty. Wszelkie oddziaływania między obiektami odbywają się za pomocą skryptów.

Ścieżki

Ścieżki służą do adresowania obiektów na różnych poziomach hierarchii filmu. Za ich pomocą możesz przesłać polecenie ze skryptu jednego obiektu do innego obiektu w filmie. Ścieżki działają na podobnych zasadach jak ścieżki dostępu w systemach plików, adresach URL i języku JavaScript. Położenie obiektu na wybranym poziomie określamy względem innego obiektu[2] lub względem głównego poziomu filmu[3]. We wcześniejszym przykładzie z **Zoo**, obiekty były ułożone na trzech poziomach:

[2] Tę metodę nazywamy adresowaniem względnym – *przyp. tłum.*

[3] Tę metodę nazywamy adresowaniem bezwzględnym – *przyp. tłum.*

- **Zoo** – główna listwa czasowa;
- **Lion** (lew) – obiekt na głównej listwie czasowej, posiadający również własną listwę czasową;
- **Teeth** (ząb) – obiekt na listwie czasowej obiektu **Lion**, posiadający również własną listwę czasową.

We Flashu 5 działają dwie konwencje adresowania. Nadal działa stara konwencja, używana we Flashu 4. Aby zaadresować obiekt **Teeth**, rozpoczynamy od głównej listwy czasowej, a następnie przechodzimy do listwy czasowej obiektu **Lion**, by wreszcie przejść do listwy czasowej obiektu **Teeth**. Adres ten wygląda tak:

```
/Lions/Teeth;
```

Jest to adres absolutny, czyli odnoszący się do obiektu względem głównego poziomu filmu. Wskazuje on, że obiekt **Teeth** znajduje się na poziomie **Lion**, który znajduje się na poziomie **Zoo**. Ponieważ poziom **Zoo** jest poziomem głównym, nie musimy podawać jego nazwy, wystarczy rozpocząć adres od ukośnika (znaku /).

Aby zaadresować obiekt, który jest nadrzędny (jak na przykład obiekt **Lion** dla obiektu **Teeth**), używamy symbolu **../** wskazującego, że odnosimy się do wyższego poziomu hierarchii. Aby przejść do głównego poziomu **Zoo** z klipu filmowego **Teeth** zagnieżdżonego w klipie filmowym **Lion**, możemy użyć adresu:

```
../../;
```

Każdy symbol **../** przechodzi o jeden poziom wyżej w hierarchii.

We Flashu 5 wprowadzono konwencję adresowania, która jest prostsza i bardziej zbliżona do konwencji stosowanych w językach programowania zorientowanych obiektowo. Szczytem hierarchii jest zawsze główna listwa czasowa. W starej konwencji adresowania sam znak / na początku adresu wystarczał jako odniesienie do głównego poziomu filmu. W nowej konwencji główną listwę czasową reprezentuje specjalna nazwa **_root**. W naszym przykładzie **Zoo** jest główną listwą czasową, więc w nowej konwencji to jej odpowiada nazwa **_root**. Adres obiektu **Teeth** wygląda następująco:

```
_root.Lions.Teeth;
```

Oprócz tego zmienne, metody i właściwości są częściami obiektu. Po zaadresowaniu obiektu, możemy odnieść się do jednego z jego elementów. Na przykład ząb lwa (obiekt **Teeth** wewnątrz obiektu **Lion**) może przy kłapnięciu wydawać dźwięk o nazwie **sound**. Adres dźwięku będzie wyglądał tak:

```
_root.Lions.Teeth.sound;
```

W dalszej części książki wielokrotnie użyjemy tej metody adresowania, szczególnie w przypadku tworzenia złożonych filmów ze skryptami. Warto więc, byś nauczył się biegle posługiwać tą metodą. Dzięki temu, gdy napotkasz skrypt ze złożoną, wielopoziomową ścieżką adresującą zmienną, właściwość lub obiekt od razu zrozumiesz, jak jest zbudowana hierarchia i do jakiego elementu się odnosi.