helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

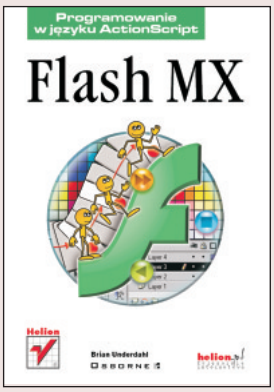
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

Flash MX.
Programowanie w języku ActionScript



Autor: Brian Underdahl
Tłumaczenie: Wojciech Pazdur
ISBN: 83-7197-950-9
Tytuł oryginału: Macromedia Flash MX: The Complete Reference
Format: stron: 252
Przykłady na ftp: 2275 kB

Pomimo prostoty język ActionScript posiada ogromne możliwości i omówienie ich wszystkich na łamach jednej książki musiałoby polegać na napisaniu opasłego tomiska, wypełnionego encyklopedyczną (a zatem mało inspirującą) wiedzą. Niniejszy podręcznik ma inny cel – przedstawienie najważniejszych i najbardziej praktycznych aspektów pracy ze skryptami oraz zachęcenie Czytelnika do samodzielnych eksperymentów.

Aby w pełni zrozumieć i bez problemu wykorzystać informacje zawarte w książce, musisz posiadać podstawowe umiejętności w posługiwaniu się Flashem. Nie oznacza to, że książka jest adresowana do ekspertów. Wystarczy, że orientujesz się w podstawowych narzędziach graficznych i edycyjnych Flasha, a także znasz najważniejsze zasady tworzenia i publikowania filmów w formacie SWF. Dzięki tej książce każdy użytkownik wcześniejszych wersji programu może bez problemu kontynuować naukę pracy z Flashem. Jeśli jednak jesteś początkującym twórcą stron internetowych, najlepiej zacząć naukę od książek „Flash MX. Od podstaw" oraz „Flash MX. Głębsze spojrzenie", wydanych nakładem Wydawnictwa Helion.

Książka skupia się na najważniejszych narzędziach i technikach pracy z językiem ActionScript. Pierwszych pięć rozdziałów powinien przeczytać każdy, niezależnie od stanu swojej wiedzy na temat skryptów Flasha. Rozdziały te nie tylko omawiają podstawowe narzędzia do tworzenia skryptów, lecz także przedstawiają szereg cennych wskazówek na temat rozwiązywania różnego rodzaju problemów dotyczących działania własnoręcznie pisanych programów. Dalsze rozdziały stanowią ilustrowane licznymi przykładami kompendium wiedzy na temat poszczególnych elementów języka ActionScript. Początkujący użytkownik Flasha powinien przeczytać wszystkie te rozdziały po kolei, natomiast osoby dobrze znające wcześniejsze wersje ActionScriptu mogą traktować ostatnie 6 rozdziałów książki jako pomoc podręczną, do której można zajrzeć w razie potrzeby i wyszukać informacje na temat odpowiedniej funkcji czy metody.

Książka „Flash MX. Programowanie w języku ActionScript" opisuje różne aspekty pracy z językiem ActionScript. Dowiesz się:

- jakich zasad należy trzymać się podczas tworzenia skryptów;
- jak korzystać z narzędzi do edycji i testowania skryptów, w szczególności z paneli Actions i Debugger;
- w jakich sytuacjach skrypty są najbardziej potrzebne;
- jak zapewnić zgodność skryptu z różnymi wersjami odtwarzaczy Flasha;
- czym są akcje, operatory, funkcje, właściwości, obiekty oraz komponenty i jakie występują między nimi powiązania.

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

Spis treści

O Autorze **11**

Wstęp **13**

Do kogo adresowana jest ta książka? 13
Jak czytać tę książkę? 13
Co można znaleźć w tej książce? 14
Przykładowe pliki 14

Rozdział 1. ActionScript — podstawy **15**

Podstawowe pojęcia związane z ActionScriptem 15
Porównanie języków ActionScript i JavaScript 16
Co to jest programowanie zorientowane obiektowo? 17
Uwaga na wielkość liter! 17
Sposób działania programu w języku ActionScript 18
Składnia języka ActionScript 18
Funkcje (functions) 19
Akcje (actions) 21
Metody (methods) 24
Właściwości (properties) 24
Zmienne (variables) 27
Stałe (constants) 29
Wyrażenia (expressions) 32
Operatory (operators) 33
Pętle (loops) 35
Komentarze (comments) 36
Znaczenie obiektów w języku ActionScript 37
Wykorzystanie notacji kropkowej (dot notation) 38
Adresowanie obiektów 39
Typy danych 41
Łańcuchy znaków (strings) 41
Liczby 42
Wartości logiczne (boolowskie) 43
Obiekty 43
Korzystanie z panelu Actions 44

Rozdział 2. Tworzenie skryptów **47**

Planowanie programu 47
Przeznaczenie programu 48
Planowanie kolejnych etapów działania 50

Dodawanie skryptów do obiektów i klatek 51

Gdzie umieścić skrypt? ... 51
Praca z panelem Actions ... 53
Różnice pomiędzy trybami Normal i Expert ... 54
Wykorzystanie ActionScriptu do sterowania właściwościami obiektów ... 57
Czym są właściwości obiektu? ... 57
Dostępne właściwości obiektów ... 58
Modyfikowanie właściwości obiektu ... 63
Stosowanie akcji w języku ActionScript ... 64
Czym są akcje? ... 64
Dostępne akcje ... 64

Rozdział 3. Przykładowe zastosowania skryptów ... 77
Wykorzystanie akcji tellTarget i with ... 77
Różnice między akcjami tellTarget i with ... 78
Sterowanie klipem filmowym przy użyciu akcji with ... 79
Interakcja programu z użytkownikiem ... 87
Typy pól tekstowych ... 88
Wysyłanie danych do serwera ... 92
Tworzenie własnego kursora ... 92
Przygotowanie klipu filmowego ... 93
Dodawanie skryptu ... 94
Dalsze możliwości ... 95

Rozdział 4. Debugger ... 97
Na czym polega proces debuggingu? ... 97
Wykorzystanie debuggera ... 99
Praca z oknem Debugger ... 99
Uruchamianie debuggera ... 100
Praca z oknem Output ... 101
Komunikaty o błędach ... 101
Lista obiektów ... 103
Lista zmiennych ... 103
Śledzenie wyrażeń ... 104
Zdalny debugging filmu ... 105

Rozdział 5. Problem z wersjami Flasha ... 107
Kto jest adresatem filmu? ... 107
Udostępnianie filmu w Internecie ... 108
Udostępnianie filmu w intranecie ... 108
Tworzenie filmów przeznaczonych do wyświetlania na platformie Pocket PC ... 109
Niezalecane instrukcje ActionScript ... 109
Kiedy można używać niezalecanych instrukcji? ... 110
Które instrukcje nie są zalecane? ... 110

Wykrywanie wersji odtwarzacza przy użyciu skryptu ... 111
Różnice pomiędzy przeglądarkami Internet Explorer i Netscape Navigator ... 111

Sprawdzanie numeru wersji ... 112
Wykorzystanie numeru wersji ... 113
Unikanie problemów z różnymi wersjami odtwarzaczy ... 114
Wykrywanie obecności odtwarzacza w systemie ... 114
Tworzenie alternatywnych wersji filmu ... 115

Rozdział 6. Akcje — opis metodyczny ... 117

Akcje służące do sterowania filmem (Movie Control) ... 117
goto ... 117
on ... 122
play ... 124
stop ... 124
stopAllSounds ... 125
Akcje związane z przeglądarką i siecią (Browser/Network) ... 126
fscommand ... 126
getURL ... 128
loadMovie ... 130
loadVariables ... 132
unloadMovie ... 133
Akcje służące do sterowania klipami filmowymi (Movie Clip Control) ... 134
duplicateMovieClip ... 134
onClipEvent ... 136
removeMovieClip ... 137
setProperty ... 137
startDrag ... 138
stopDrag ... 139
updateAfterEvent ... 139
Akcje związane ze zmiennymi (Variables) ... 140
delete ... 140
set variable ... 140
var ... 141
with ... 141
Pętle i instrukcje warunkowe (Condition/Loops) ... 142
break ... 142
case ... 143
continue ... 143
default ... 143
do..while ... 143
else ... 144
else if ... 145
for ... 145
for..in ... 147
if ... 148
switch ... 149
while ... 149

Akcje związane z drukowaniem (Printing) ... 149
print ... 150
Funkcje definiowane przez użytkownika (User-Defined Functions) ... 152

call ... 152
call function ... 153
function ... 153
method ... 155
return ... 156
Akcje różne (Miscellaneous) ... 156
#endinitclip ... 156
#include ... 157
#initclip ... 158
clearInterval ... 158
comment ... 158
evaluate ... 159
setInterval ... 159
trace ... 160
Akcje niezalecane ... 160
ifFrameLoaded ... 160
tellTarget ... 161
toggleHighQuality ... 162

Rozdział 7. Operatory — opis metodyczny ... 163

Operatory podstawowe ... 163
" " (cudzysłowy) ... 163
() (nawiasy) ... 164
Operatory arytmetyczne (Arithmetic Operators) ... 164
% (reszta z dzielenia) ... 164
* (mnożenie) ... 165
+ (dodawanie) ... 165
- (odejmowanie) ... 165
/ (dzielenie) ... 165
Operatory przypisania (Assignment) ... 166
%= (przypisanie reszty z dzielenia) ... 166
&= (przypisanie bitowego iloczynu logicznego — AND) ... 166
|= (przypisanie bitowej sumy logicznej — OR) ... 166
*= (przypisanie iloczynu) ... 167
+= (przypisanie sumy) ... 167
-= (przypisanie różnicy) ... 167
/= (przypisanie ilorazu) ... 167
<<= (przypisanie bitowego przesunięcia w lewo) ... 167
= (przypisanie) ... 167
>>= (przypisanie bitowego przesunięcia w prawo) ... 167
>>>= (przypisanie bitowego przesunięcia w prawo bez zachowania znaku) ... 168
^= (przypisanie bitowej różnicy symetrycznej — XOR) ... 168
Operatory poziomu bitowego (Bitwise Operators) ... 168
& (bitowy iloczyn logiczny — AND) ... 169
~ (bitowa negacja logiczna — NOT) ... 169
| (bitowa suma logiczna — OR) ... 169
<< (przesunięcie bitów w lewo) ... 170
>> (przesunięcie bitów w prawo) ... 170
>>> (przesunięcie bitów w prawo bez zachowania znaku) ... 170

^ (bitowa różnica symetryczna — XOR) 170
Operatory porównania (Comparison Operators) 171
!= (nierówność) 171
< (mniejsze) 171
<= (mniejsze lub równe) 172
= = (równość) 172
= = = (równość i zgodność typów) 172
> (większe) 172
>= (większe lub równe) 173
Operatory logiczne (Logical Operators) 173
! (negacja logiczna — NOT) 173
&& (iloczyn logiczny — AND) 174
|| (suma logiczna — OR) 174
Operatory różne (Miscellaneous Operators) 174
++ (inkrementacja) 175
-- (dekrementacja) 175
?: (warunek) 175
instanceof 176
typeof 176
void 176
Operatory niezalecane 177
<> (nierówność) 177
add 177
and 177
eq 178
ge 178
gt 178
le 178
lt 178
ne 178
not 178
or 178
Operatory dodatkowe 179
, (przecinek) 179
. (kropka) 179
// (linia komentarza) 179
/* (blok komentarza) 179
[ ] (operator dostępu do tablicy) 180
{ } (operator inicjalizacji obiektu) 180

Rozdział 8. Funkcje — opis metodyczny 183
Funkcje podstawowe 183
escape 183
eval 184
getProperty 184
getTimer 185
getVersion 186
targetPath 186
unescape 186

Funkcje konwertujące (Conversion Functions) ... 187
Array ... 187
Boolean ... 187
Number ... 188
Object ... 188
String ... 189
Funkcje matematyczne (Mathematical Functions) ... 189
isFinite ... 189
isNaN ... 189
parseFloat ... 190
parseInt ... 191
Funkcje niezalecane ... 191
chr ... 192
int ... 192
length ... 192
mbchr ... 193
mblength ... 193
mbord ... 193
mbsubstring ... 193
ord ... 194
random ... 194
substring ... 194

Rozdział 9. Właściwości — opis metodyczny ... 197

_alpha ... 197
_currentframe ... 198
_droptarget ... 199
_focusrect ... 199
_framesloaded ... 200
_height ... 200
_highquality ... 201
_name ... 202
_parent ... 203
_quality ... 203
_root ... 204
_rotation ... 204
_soundbuftime ... 205
_target ... 205
_totalframes ... 206
_url ... 206
_visible ... 206
_width ... 206
_x ... 207
_xmouse ... 208
_xscale ... 208
_y ... 209
_ymouse ... 209

_yscale ... 209

Rozdział 10. Obiekty — opis metodyczny ... 211

Obiekty ActionScript ... 211
Obiekty podstawowe (Core) ... 212
Obiekty związane z filmem (Movie) ... 212
Obiekty związane z komunikacją klient-serwer (Client/Server) ... 212
Obiekty dodatkowe (Authoring) ... 213
Array ... 213
Array.concat ... 214
Array.join ... 214
Array.length ... 214
Array.pop ... 215
Array.push ... 215
Array.reverse ... 215
Array.shift ... 215
Array.slice ... 216
Array.sort ... 216
Array.splice ... 216
Array.toString ... 217
Array.unshift ... 217
Color ... 217
Color.setRGB ... 217
Color.getRGB ... 218
Color.setTransform ... 218
Color.getTransform ... 219
Date ... 219
Key ... 219
Metody i stałe obiektu Key ... 219
Wykorzystanie obiektu Key ... 221
Math ... 222
Metody i stałe obiektu Math ... 222
Wykorzystanie obiektu Math ... 222
Mouse ... 224
MovieClip ... 224
Number ... 224
Selection ... 226

Sound ... 227
Sound.attachSound ... 227
Sound.getBytesLoaded ... 227
Sound.getBytesTotal ... 227
Sound.getPan ... 227
Sound.getTransform ... 228
Sound.getVolume ... 228
Sound.loadSound ... 228
Sound.setPan ... 228
Sound.setTransform ... 228
Sound.setVolume ... 229

Sound.start ... 229
Sound.stop ... 229
String ... 229
String.charAt ... 230
String.charCodeAt ... 230
String.concat ... 230
String.fromCharCode ... 230
String.indexOf ... 230
String.lastIndexOf ... 230
String.length ... 231
String.slice ... 231
String.split ... 231
String.substr ... 231
String.substring ... 231
String.toLowerCase ... 231
String.toUpperCase ... 232
XML ... 232
XMLSocket ... 232

Rozdział 11. Wykorzystanie gotowych komponentów ... 235
Czym są komponenty? ... 235
Zastosowania komponentów ... 235
Typy komponentów ... 236
Wstawianie komponentów do filmu ... 238
Planowanie użycia komponentów ... 238
Wstawianie komponentów do sceny ... 239
Komponenty a ActionScript ... 240
Przykład wykorzystania komponentów ... 240

Skorowidz ... 245

Rozdział 3.
Przykładowe zastosowania skryptów

Animacje możesz tworzyć bez wykorzystania ActionScriptu — na przykład za pomocą narzędzi *Motion Tween* czy *Shape Tween*. Jednak możliwości tych narzędzi są czasami niewystarczające i ani jedno, ani drugie nie pozwala uzyskać zamierzonego efektu. W takich sytuacjach warto zaprzęgnąć do pracy ActionScript, za pomocą którego można osiągnąć niemal każdy zamierzony cel.

Trzeci rozdział niniejszej książki poświęcę kilku przykładom różnych ciekawych efektów, których uzyskanie nie byłoby możliwe bez użycia języka ActionScript. Czytając niniejszy rozdział, nauczysz się korzystać z akcji `tellTarget` i `with` w celu sterowania klipami filmowymi, a także dowiesz się, jak umożliwić interakcje filmu z użytkownikiem.

Tytuł rozdziału mógłby sugerować, że od razu rzucamy się na głęboką wodę i z marszu zajmujemy się programowaniem. W rzeczywistości mam zamiar raczej uświadomić Ci kilka możliwości skryptów Flash i przekonać Cię, jak prosty i wygodny w użyciu jest język ActionScript. Chociaż zadania omawiane w tym rozdziale mogą początkującemu Czytelnikowi wydać się nieco trudne, wykonanie ich z pewnością pomoże zrozumieć wiele aspektów pracy ze skryptami Flasha. Być może po przeczytaniu rozdziału dojdziesz do wniosku, że tworzenie różnych efektów przy użyciu skryptów jest znacznie prostsze, niż Ci się wydawało!

Wykorzystanie akcji tellTarget i with

W niemal wszystkich filmach Flasha występują różnego rodzaju animacje. Czasami do ich utworzenia wystarczą standardowe narzędzia *Motion Tween* i *Shape Tween*, jednak często pojawia się potrzeba wykorzystania bardziej zaawansowanych technik.

Klipy filmowe (*Movie Clips*) Flasha stanowią jeden z najciekawszych typów obiektów Flasha. W istocie klipy same w sobie są minifilmami, które program wyświetla w obrębie filmu głównego. Pozwala to na tworzenie wielu rozmaitych efektów, których nie można byłoby uzyskać w wyniku animacji innych typów obiektów.

Po umieszczeniu w scenie klonu (*instance*) klipu filmowego często chcemy mieć możliwość sterowania tym klipem z poziomu innych obiektów w filmie. Jako przykład rozważ zestaw przycisków sterujących odtwarzaniem zawartości klipu filmowego. Jeden z przycisków ma włączać odtwarzanie, drugi zatrzymywać, trzeci przewijać klip do początku itd. Aby zrealizować takie rozwiązanie, musisz określić sposób komunikowania się przycisków

z klipem. Choć sama idea nie wydaje się zbyt skomplikowana, mechanizmy kryjące się za tym procesem są bardziej złożone, niż mogłoby się wydawać. Weź pod uwagę kilka problemów, które będziesz musiał rozwiązać podczas pisania programu:

- Ponieważ przyciski nie komunikują się bezpośrednio z klipem filmowym, musisz zdefiniować mechanizm przekazywania danych do klipu, aby ten zareagował odpowiednio na użycie każdego z tych przycisków.
- Może być konieczne pobieranie pewnych dodatkowych informacji z klipu filmowego, na przykład na temat tego, czy odtwarzanie klipu nie zostało zakończone.
- W filmie mogą znajdować się równocześnie różne klipy filmowe, dlatego należy zapewnić, aby dane z przycisków były adresowane tylko do właściwego obiektu (a nie do np. każdego klipu w filmie).

Jak zapewne się domyśliłeś, Flash udostępnia efektywne narzędzia umożliwiające rozwiązanie każdego z przedstawionych wyżej problemów. W większości przypadków narzędzia te mają postać odpowiednio zaimplementowanych właściwości (*properties*) obiektów.

Zanim jednak przejdziemy do szczegółów, zwróćmy uwagę na kilka ważnych spraw związanych z wykorzystaniem klonów symboli w filmach Flasha:

- Gdy używasz jakiegoś symbolu w filmie, do sceny wstawiany jest klon (*instance*) tego symbolu, a nie sam symbol. Jest to o tyle ważne, że ma duży wpływ na objętość pliku publikowanego w sieci. Niezależnie od tego, ilu klonów danego symbolu użyjesz w filmie, symbol ten zajmie w pliku tyle samo miejsca.
- Klonom symboli trzeba nadawać nazwy, jeśli chcemy powiązać je z innymi obiektami w kodzie ActionScript.
- Właściwości każdego klonu możesz modyfikować niezależnie od pozostałych klonów.
- Zmiany wprowadzone w symbolu znajdującym się w bibliotece zostają odzwierciedlone na wszystkich klonach tego symbolu umieszczonych w scenie.

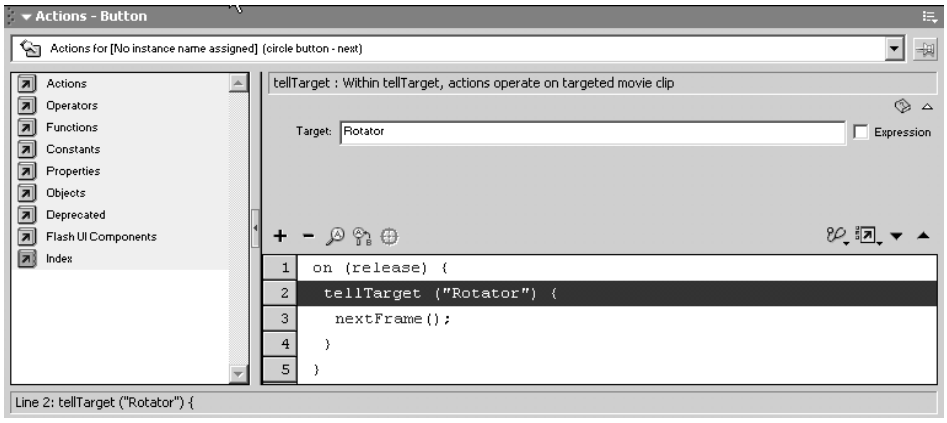
Do wysyłania i pobierania danych z klonów klipu filmowego możesz posłużyć się dwiema akcjami: `tellTarget` i `with` (nie są to jedyne akcje umożliwiające pracę z klipami filmowymi, jednak dla potrzeb omawianego tu przykładu są one najbardziej użyteczne).

Różnice między akcjami tellTarget i with

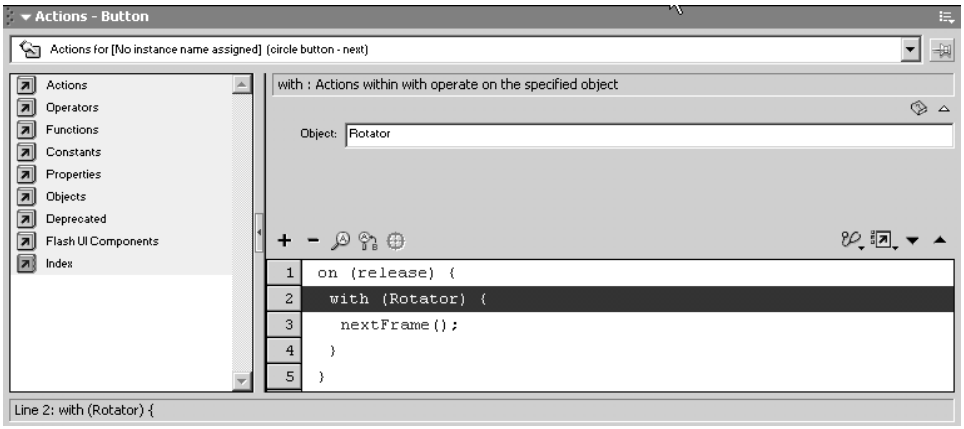
Wiele wcześniejszych publikacji na temat Flasha wypełnionych jest przykładami dotyczącymi wykorzystania akcji `tellTarget`, natomiast akcja `with` jest w nich często pomijana. Mógłbyś dzięki temu nabrać przekonania, że instrukcja `tellTarget` jest w czymś lepsza od instrukcji `with`. W rzeczywistości tak nie jest.

Akcja `tellTarget` znajduje się obecnie na liście tych akcji, których stosowanie nie jest zalecane i być może w następnych wersjach Flasha nie będzie już obsługiwana. Dla odmiany, akcja `with` nie tylko jest w pełni obsługiwana przez najnowsze wersje programu (Flash 5 i najnowsza — Flash MX), ale także stanowi narzędzie o wiele bardziej uniwersalne niż akcja `tellTarget`. Działanie akcji `tellTarget` ogranicza się bowiem wyłącznie do klipów filmowych, natomiast akcji `with` możemy używać z dowolnymi typami obiektów.

Podczas kodowania nie występują praktycznie żadne znaczące różnice pomiędzy obydwiema omawianymi akcjami. Możesz się o tym przekonać, porównując dwa poniższe obrazki. Na obu z nich widać kod skryptu, którego zadaniem jest przejście do następnej klatki animacji w obrębie klipu filmowego o nazwie `"Rotator"`. W pierwszym przypadku posłużono się do tego celu akcją `tellTarget`.



Drugi rysunek przedstawia ten sam skrypt w wersji z akcją `with`.





Chociaż obydwie wersje skryptu wyglądają niemal identycznie, występuje między nimi pewna różnica, na którą trzeba zwrócić uwagę. Zauważ, że nazwa klonu klipu filmowego w akcji `tellTarget` podawana jest w cudzysłowach, natomiast w akcji `with` — bez cudzysłowów. Różnicę tę musisz uwzględnić, gdy np. próbujesz zaadaptować kod pochodzący z napisanych dawniej skryptów do najnowszej wersji Flasha.

Sterowanie klipem filmowym przy użyciu akcji with

Aby przedstawić praktyczne aspekty wykorzystania akcji `with`, zaprezentuję proces tworzenia przykładowego programu w języku ActionScript. Przykład ten jest o tyle cenny, że jego realizacja innymi metodami byłaby dość trudna i czasochłonna.

Załóżmy, że chcemy zareklamować i sprzedać pewien produkt, który należałoby pokazać klientom w kilku różnych widokach. Naszym celem będzie zatem stworzenie takiej aplikacji, w której widz przy użyciu odpowiednich przycisków może obracać widok w lewo lub w prawo. Ogólnie rzecz biorąc, chodzi o to, aby widz miał odczucie, że produkt znajduje się na obrotowym podeście, którym można poruszać, naciskając przyciski na pulpicie sterowniczym.

Tworzenie klipu filmowego

Klipy filmowe można tworzyć różnymi metodami. W tym przypadku zakładamy, że chcemy zareklamować samochód i umożliwić widzowi obejrzenie go z różnych stron. Wykorzystamy do tego celu serię zdjęć samochodu, które przedstawiają pojazd pod różnymi kątami. Zaczniemy od przygotowania obrazków ze zdjęciami.

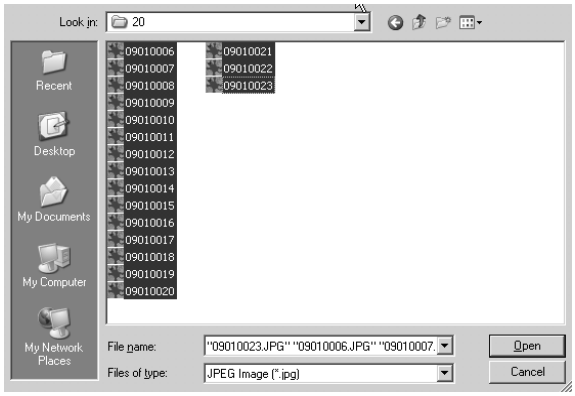
Kolejne fotografie powinny być wykonane z pozycji mniej więcej równomiernie rozmieszczonych wokół samochodu. Ważne jest też zachowanie tej samej odległości od środka pojazdu i wykonanie tylu zdjęć, aby możliwe było dokładne pokazanie wszystkich jego elementów. Liczba zdjęć, które należy wykonać, zależna jest oczywiście od stopnia złożoności obiektu, więc w przypadku filmów przedstawiających inne produkty może być potrzebna mniejsza lub większa liczba fotografii niż w omawianym przykładzie.



Jeśli chcesz umożliwić obracanie widoku produktu, pamiętaj, aby odpowiednio ponumerować kolejne fotografie. Dzięki temu łatwo będziesz mógł rozmieścić je w kolejnych ujęciach klipu filmowego.

Przejdźmy zatem do rzeczy. Na początku musimy utworzyć sam klip filmowy. Oto czynności, jakie należy w tym celu wykonać:

1. Utwórz serię obrazków przedstawiających produkt. Mogą to być zeskanowane lub wykonane aparatem cyfrowym fotografie, ale pamiętaj, że rozmiary obrazków powinny być stosunkowo niewielkie. W przeciwnym razie czas przesyłania filmu siecią może być dość długi.
2. Otwórz nowy, pusty projekt Flasha.
3. Wybierz polecenie *File/Import* (lub użyj skrótu *Ctrl+R*), aby otworzyć okno dialogowe *Import*, przedstawione na rysunku poniżej.



4. Wybierz odpowiedni format plików z rozwijanej listy w dole okna. W przypadku zdjęć będzie to zapewne format *JPEG Image*.

5. Wybierz nazwy plików, które chcesz załadować. Możesz równocześnie wyróżnić całą serię plików, klikając pierwszą nazwę, przytrzymując klawisz *Shift* i klikając ostatnią nazwę w serii. Jeśli chcesz załadować kilka plików, których nazwy nie stanowią ciągłej serii na liście, przytrzymaj wciśnięty klawisz *Ctrl* i klikaj osobno każdą z nazw. Jeśli nazwy plików zawierają kolejno uporządkowane numery, wystarczy wybrać pierwszy plik z serii. Plik z gotowymi wynikami ćwiczenia (*MYCAR.FLA*) oraz pliki JPG wykorzystane w tym ćwiczeniu znajdziesz na serwerze FTP wydawnictwa „Helion", pod adresem: *ftp://ftp.helion.pl/przyklady/flmxas.zip*.

6. Kliknij przycisk *Open*, aby załadować wybrane pliki do programu. Zostaną one umieszczone i w scenie, i w bibliotece filmu (*Library*).

7. Usuń wszystkie obrazki ze sceny. W naszym przypadku konieczne będzie ich wstawienie do odpowiedniego klipu filmowego, a nie do głównej listwy czasowej filmu.

8. Wybierz polecenie *Insert/New Symbol* lub naciśnij klawisz *Ctrl+F8*, aby otworzyć okno dialogowe *Create New Symbol*, w którym definiujemy nowy symbol.

9. Wprowadź nazwę symbolu w polu *Name* (na przykład *Images*) i kliknij przycisk *OK*, aby przejść dalej.

10. Otwórz okno biblioteki (*Library*), wybierając polecenie *Window/Library* lub używając skrótu *F11*.

11. Przeciągnij pierwszy obrazek z biblioteki do pierwszej klatki na listwie czasowej nowego symbolu.

12. Kliknij następną klatkę na listwie czasowej, aby ją zaznaczyć.

13. Wybierz polecenie *Insert/Blank Keyframe* lub posłuż się klawiszem *F7*, aby wstawić pustą klatkę kluczową.

14. Przeciągnij drugi obrazek do nowej klatki kluczowej.

15. Powtarzaj czynności z punktów od 12. do 14. dla wszystkich kolejnych obrazków z biblioteki, aż każdy z znajdzie się w odpowiedniej klatce na listwie czasowej klipu. Gdy to zrobisz, możesz zamknąć okno *Library* lub odsunąć je na bok, aby nie przeszkadzało Ci w dalszej pracy nad klipem (później będziemy musieli jeszcze skorzystać z tego okna, gdy pojawi się konieczność wstawienia klipu filmowego do sceny).

16. Wyświetl panel wyrównywania obiektów (*Align*), używając polecenia *Windows/Panels/Align* lub skrótu *Ctrl+K*.

17. Kliknij w klatce nr 1, aby ją zaznaczyć. Obrazek wstawiony do tej klatki powinien być otoczony ramką informującą o tym, że zaznaczyłeś klatkę.

18. Kliknij ikonę *To Stage* w panelu *Align*, dzięki czemu obrazki będą wyrównywane względem środka klipu filmowego. Musimy wyrównać wszystkie obrazki, aby każdy z nich był wyświetlany w tym samym miejscu podczas przechodzenia z jednej klatki do drugiej.

19. Użyj ikon *Align Horizontal Center* i *Align Vertical Center* w panelu *Align* do ustawienia pozycji obrazka w pionie i w poziomie.

20. Zaznacz następną klatkę na listwie czasowej klipu i powtórz czynności z punktu 19. To samo zrób we wszystkich kolejnych klatkach, dzięki czemu wszystkie obrazki będą wycentrowane względem obszaru zajmowanego przez klip filmowy. Gdy skończysz, możesz zamknąć panel *Align*.

21. Wybierz polecenie *Insert/Layer* lub użyj ikony *Insert Layer* i dodaj dwie nowe warstwy do listwy czasowej. Pierwszą z nich możesz nazwać *Labels* (zostaną na niej umieszczone etykiety klatek), a drugiej nadaj nazwę *Actions*, ponieważ będziemy w jej obrębie definiować akcje ActionScriptu.

22. Wyświetl panel *Properties*, używając polecenia *Window/Properties* lub skrótu klawiaturowego *Ctrl+F3*.

23. Kliknij pierwszą klatkę w warstwie *Labels* i przypisz tej klatce etykietę *FirstPic*, wprowadzając ją w polu *<Frame Label>* w panelu *Properties*.

24. Zaznacz ostatnią klatkę w warstwie *Labels* (to znaczy tę, której odpowiada ostatni obrazek dodany do klipu).

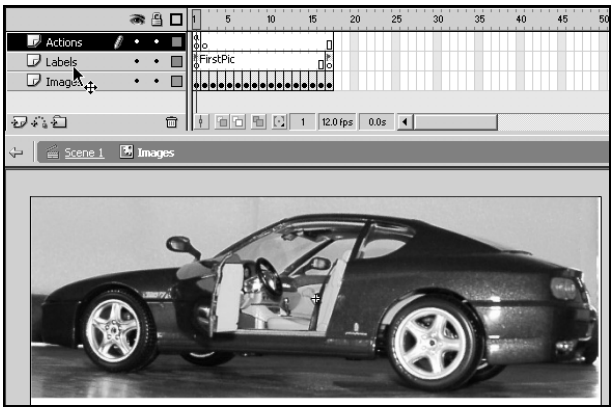
25. Wybierz polecenie *Insert/Keyframe* lub naciśnij *F6*, co spowoduje wstawienie klatki kluczowej, której można nadać odpowiednią etykietę.

26. Ponownie przejdź do panelu *Properties* i w polu *<Frame Label>* wprowadź nazwę *LastPic*. Możesz zamknąć panel *Properties*.

27. Kliknij pierwszą klatkę warstwy *Actions*.

28. Otwórz panel *Actions*, wykorzystując polecenie *Window/Actions* lub skrót klawiaturowy *F9*.

29. Z katalogu *Actions/Movie Control* po lewej stronie panelu *Actions* wybierz podwójnym kliknięciem akcję `stop`. Zostanie ona dodana do okna skryptu i tym samym odtwarzanie kolejnych klatek klipu będzie zatrzymywane zaraz po jego załadowaniu. Potrzeba takiego rozwiązania wynika z tego, że chcemy pozwolić użytkownikowi na to, aby sam przechodził do kolejnych klatek za pomocą odpowiednich przycisków. Ekran Twojego komputera powinien teraz wyglądać mniej więcej tak jak na poniższym obrazku.



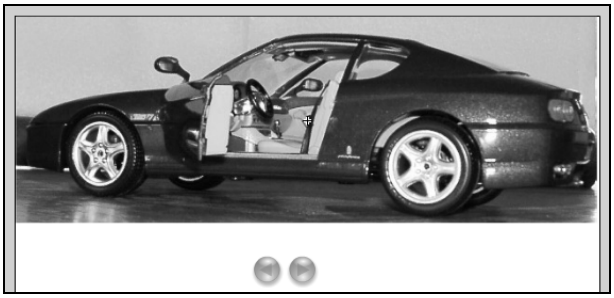
30. Kliknij nazwę sceny *Scene 1* pod listwą czasową, aby przejść do głównej listwy czasowej projektu. Choć klip filmowy nie jest teraz widoczny, został on zapisany w bibliotece i jest gotowy do użycia.

Tworzenie klipu filmowego z serii zdjęć było zadaniem dość prostym, choć momentami pewnie nużącym. Jak przekonasz się za chwilę, przygotowanie pozostałej części filmu również nie nastręczy nam żadnych kłopotów.

Dodawanie obiektów do sceny głównej

Po opracowaniu klipu filmowego możesz zająć się rozmieszczeniem odpowiednich elementów w scenie głównej. W tym celu wykonaj następujące czynności:

1. Jeśli wcześniej zamknąłeś okno biblioteki (*Library*), otwórz je ponownie.
2. Przeciągnij klon klipu filmowego *Images* z biblioteki do sceny. Ustaw go tak, aby pod klipem zostało dość miejsca na wstawienie tam przycisków.
3. Wybierz polecenie *Window/Common Libraries/Buttons*, otwierając bibliotekę z gotowymi zestawami przycisków.
4. Wyszukaj w bibliotece dwa odpowiadające Ci przyciski, jeden ze strzałką w lewo, a drugi w prawo. Ja wybrałem przyciski *gel Left* i *gel Right* z katalogu *Playback*.
5. Umieść klony każdego z wybranych przycisków w scenie i ustaw je pod klipem filmowym. Scena powinna teraz wyglądać tak jak na poniższym rysunku.



Możesz zamknąć wszystkie okna bibliotek, ponieważ nie będą one już potrzebne. Następnym ważnym zadaniem jest nadanie nazwy klonowi klipu filmowego.

Nadawanie nazwy klonowi klipu filmowego

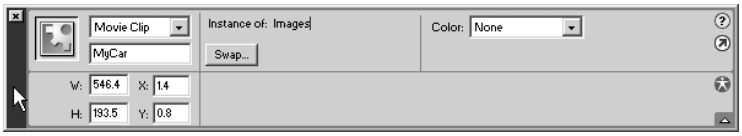
Aby możliwe było skorzystanie z akcji `with` (lub `tellTarget`), musisz przypisać niepowtarzalną nazwę klonowi klipu filmowego. Czynność ta jest niezwykle istotna dla poprawnego działania programu.

W celu nadania obiektowi nazwy wykonaj następujące czynności:

1. Wyświetl panel *Properties*, używając polecenia *Window/Properties* lub skrótu klawiaturowego *Ctrl+F3*.

2. Kliknij klon klipu filmowego, aby go zaznaczyć.

3. Wprowadź nazwę klonu w polu *<Instance Name>*. W moim przypadku (zobacz rysunek poniżej) nadałem klonowi nazwę *MyCar*. Nazwa obiektu może brzmieć dowolnie, jednak później należy pamiętać o tym, aby w kodzie skryptu wprowadzać ją w dokładnie tej samej postaci co w panelu *Properties*.



4. Zamknij panel *Properties*. W przypadku tego filmu klip jest jedynym obiektem, którego nazwa będzie wykorzystywana w skrypcie.



Nadając nazwy klonom obiektów we Flashu, używaj dwóch lub więcej krótkich słów, które nie będą się myliły z nazwami innych obiektów ani ze słowami kluczowymi ActionScript.

Wprowadzanie kodu ActionScript

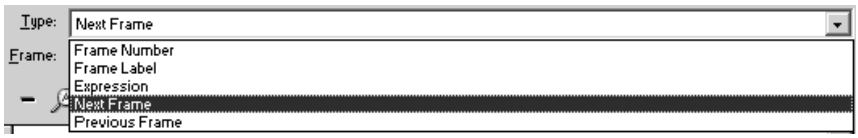
Teraz możemy już zająć się tworzeniem samego skryptu, który sprawi, że film będzie działał zgodnie z naszymi oczekiwaniami. Zanim jednak zajmiesz się tym zadaniem, dokładnie rozważ, co właściwie jest celem działania skryptu:

- Po załadowaniu filmu powinna zostać wyświetlona pierwsza klatka klipu i na niej odtwarzacz powinien się zatrzymać. To założenie już spełniliśmy, wstawiając akcję `stop` do listwy czasowej klipu.
- Gdy użytkownik kliknie przycisk ze strzałką skierowaną w prawo, wskaźnik czasu powinien przejść do następnej klatki na listwie czasowej klipu.
- Gdy użytkownik kliknie przycisk ze strzałką w lewo, wskaźnik czasu powinien przesunąć się o jedną klatkę do tyłu.
- Jeśli wskaźnik czasu dojdzie do pierwszej lub ostatniej klatki filmu, kolejne kliknięcie przycisku powinno spowodować przeskok do przeciwnego końca listwy czasowej.

Ponieważ pierwsze zadanie zostało już zrealizowane, musimy zająć się trzema kolejnymi punktami. Zacznijmy od tego, w jaki sposób przemieszczać wskaźnik czasu w lewo lub w prawo.

Jeśli przyjrzysz się akcjom z katalogu *Actions* w oknie *Actions*, zauważysz zapewne, że żadna z nich nie pozwala bezpośrednio zrobić tego, o co nam chodzi. I co z tym fantem począć? Mam dla Ciebie niespodziankę — nie wszystkie akcje dostępne w języku ActionScript można znaleźć w katalogach znajdujących się w tym panelu. Możesz się o tym przekonać, przeglądając słowniczek ActionScript Dictionary, otwierany poleceniem *Help/ActionScript Dictionary*. Tak czy inaczej, rozwiązanie naszego problemu wymaga pewnych specjalnych zabiegów.

W celu zrealizowania założeń przedstawionych na powyższej liście skorzystamy z opcji dostępnych dla akcji `gotoAndStop` (akcja ta znajduje się katalogu *Movie Control*). Jak przedstawia poniższy rysunek, po rozwinięciu listy *Type* w polu z parametrami akcji możesz wybrać jedną z kilku opcji, w tym *Next Frame* i *Previous Frame*. Jeśli wybierzesz jedną z tych dwóch pozycji, możesz być zdziwiony tym, że zamiast akcji `gotoAndStop` w oknie skryptu pojawi się zapis `nextFrame()` lub `prevFrame()`.



Jak wspomniałem, trzeba odpowiednio podejść do sytuacji, w której wskaźnik czasu dojdzie do lewego lub prawego końca zakresu klatek. Kod ActionScript zostanie umieszczony wewnątrz przycisków i będzie komunikował się z klipem filmowym w celu określenia, która klatka wyświetlana jest w danej chwili. Na podstawie tych danych podejmie decyzję o przejściu do sąsiedniej klatki lub do klatki z drugiego końca zakresu. Właśnie z tego powodu nadawaliśmy etykiety *FirstPic* i *LastPic*, odpowiednio: pierwszej i ostatniej klatce klipu. Nazwę klipu *MyCar* podamy w wywołaniu akcji `with`, aby móc zarówno odczytywać właściwości klipu filmowego, jak i je modyfikować.

Skrypt umieszczony w obrębie przycisku ze strzałką w lewo powinien wyglądać następująco:

```
on (release) {
    with (MyCar) {
            if (_currentframe == 1) {
                    gotoAndStop("LastPic");
            } else {
                    prevFrame();
            }
        }
}
```

Przeanalizujmy kolejne jego linie. Pierwsza z nich to:

```
on (release) {
```

W linii tej umieszczono uchwyt zdarzenia `release`, co oznacza, że wykrywane jest zwolnienie przycisku myszy po kliknięciu. Linia ta została dodana automatycznie w trakcie wprowadzania akcji `with`, którą program umieścił w drugiej linii kodu:

```
    with (MyCar) {
```

Akcja `with` odwołuje się do klipu filmowego *MyCar* i rozpoczyna blok kodu, w którym zapisane są instrukcje dotyczące tego klipu. Pierwszą z nich jest instrukcja warunkowa `if`:

```
            if (_currentframe == 1) {
                    gotoAndStop("LastPic");
```

Sprawdzane w niej jest, czy bieżącą klatką (`_currentframe`) nie jest pierwsza klatka klipu. Gdy wystąpi taka sytuacja, oznacza to, że wskaźnik czasu znajduje się na lewym końcu zakresu i nie można go przesuwać dalej w tę stronę (właściwość `_currentframe` zwraca

zawsze wartość liczbową, dlatego nie mogliśmy tu użyć etykiety klatki zamiast numeru). Jeśli wynikiem instrukcji `if` będzie wartość true, wskaźnik bieżącej klatki na listwie czasowej zostanie przesunięty na przeciwny koniec zakresu, czyli do klatki z etykietą `"LastPic"`. Jeśli operacja ta zostanie wykonana, program będzie czekał na dalsze akcje ze strony użytkownika. Jeśli natomiast nie było konieczne przejście na drugi koniec zakresu, wskaźnik czasu zostanie po prostu przesunięty o jedną pozycję w lewo:

```
            } else {
                    prevFrame();
```

Inaczej mówiąc, wskaźnik czasu przesuwany jest o jedną klatkę do tyłu, pod warunkiem że nie znajduje się w pierwszej klatce klipu.

Zwróć uwagę na to, że nie musieliśmy używać nazwy klipu filmowego w żadnej linii oprócz tej z definicją akcji `with`. Jest to wynikiem użycia tej akcji, która sprawia, że wszystkie instrukcje w obrębie bloku kodu adresowane są do klipu wyszczególnionego na początku. Dzięki użyciu akcji `with` Flash wie, że wszystkie kolejne akcje w bloku dotyczą klipu o nazwie *MyCar*.

Jak zapewne się domyśliłeś, kod umieszczony w przycisku ze strzałką w prawo powinien być podobny do tego w omówionym przypadku. Jego zapis przedstawia się następująco:

```
on (release) {
    with (MyCar) {
            if (_currentframe == 17) {
                    gotoAndStop("FirstPic");
            } else {
                    nextFrame();
            }
        }
}
```

Przypominam, że aby wstawić skrypt do przycisku, należy zaznaczyć ten przycisk, otworzyć panel *Actions* (polecenie *Window/Actions* lub skrót klawiaturowy *F9*), a następnie wybierać podwójnymi kliknięciami kolejne instrukcje z katalogów znajdujących się po lewej stronie panelu. W przypadku niektórych z nich konieczne jest wypełnienie pól z parametrami, widocznych ponad oknem skryptu.



Gdy wprowadzasz kod skryptu, nie zapomnij użyć znaku podkreślenia (_) na początku nazwy właściwości `_currentframe`. Pamiętaj o tym, że wszystkie nazwy właściwości obiektu rozpoczynają się tym znakiem. Jeśli o nim zapomnisz, Flash będzie traktował wpisaną przez Ciebie nazwę jako nazwę jakiejś zmiennej, a nie właściwości obiektu.

Testowanie filmu

Nadeszła chwila prawdy. Czy to wszystko zadziała? Aby się o tym przekonać, użyj polecenia *Control/Test Movie* lub naciśnij klawisze *Ctrl+Enter*. Poniżej przedstawiam dwie klatki z mojej wersji filmu.



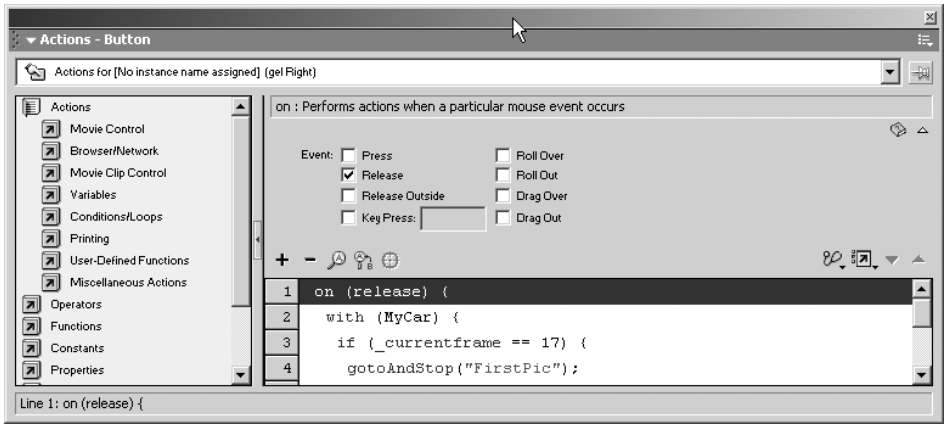


Gdy będziesz testował film, zwróć szczególną uwagę na to, co dzieje się po dojściu do pierwszej lub ostatniej klatki klipu. Czy następuje wtedy przeskok do klatki po przeciwnej stronie zakresu? Jeśli nie, wróć do skryptów zdefiniowanych w przyciskach. Wszystkie nazwy akcji powinny być wyświetlane na niebiesko, właściwości na zielono, a etykiety, zmienne oraz wartości liczbowe na czarno. Błędy składniowe Flash wyróżnia na czerwono, ale najczęściej to nie one są przyczyną problemów z działaniem filmu, a program zazwyczaj nie może pomóc Ci w wykryciu innych błędów. Jeśli nie potrafisz znaleźć błędów w swoich skryptach, porównaj je dokładnie z listingami wydrukowanymi powyżej (lub ze skryptami zawartymi w pliku *MYCAR.FLA*, który znajdziesz w zbiorze materiałów do niniejszej książki, pod adresem: *ftp://ftp.helion.pl/przyklady/flmxas.zip*).

Interakcja programu z użytkownikiem

W przykładzie z poprzedniego podrozdziału interakcja filmu i użytkownika była bardzo prosta. Gdy użytkownik klikał jeden z dwóch przycisków, wskaźnik na listwie czasowej klipu przesuwał się w lewo lub w prawo. Pojedynczy uchwyt zdarzenia `on (release)` realizował całą komunikację z użytkownikiem, który mógł jedynie kliknąć pierwszy lub drugi przycisk.

Po wykonaniu poprzedniego ćwiczenia powinieneś już rozumieć, jak działają uchwyty zdarzeń związanych z obsługą myszy. Lista zdarzeń (*events*), które mogą być wykrywane przez program, jest dość bogata, co przedstawia poniższy rysunek.



Czasami konieczne jest jednak zaoferowanie użytkownikowi innych możliwości wprowadzania danych niż tylko klikanie myszą. Często pojawia sytuacja, w której chcemy, aby widz wprowadził pewne informacje do formularza i wysłał je do serwera sieciowego. Zajmiemy się teraz kilkoma przykładami tego typu rozwiązań.

Typy pól tekstowych

W dynamicznych polach tekstowych użytkownik może wprowadzać dane z klawiatury. Możesz na przykład poprosić go o podanie imienia, a później wyświetlać imię w komunikatach adresowanych do widza.

W niniejszym przykładzie użyjemy pól tekstowych Flasha do zrealizowania prostego mechanizmu komunikacji użytkownika z programem. Dowiesz się przy tym, jak możesz sprawdzić, czy widz wprowadził swoje imię, zanim nastąpi przejście do dalszej części filmu.

Definiowanie pól tekstowych i zmiennych

We Flashu można korzystać z trzech typów pól tekstowych i każdego z nich użyjemy w tym przykładzie. Na początku omówię pokrótce każdy z dostępnych typów pól tekstowych:

- *Static Text* — statyczne pole tekstowe. Zawiera napisy, które nie zmieniają się w trakcie wyświetlania filmu, na przykład objaśnienia do przycisków lub opcji.
- *Dynamic Text* — dynamiczne pole tekstowe. Jego zawartość może zmieniać się w trakcie odtwarzania filmu.
- *Input Text* — wejściowe pole tekstowe. Pozwala wprowadzać widzowi tekst podczas odtwarzania filmu.

Zajmiemy się teraz prostym filmem, który będzie zawierał tylko dwie klatki. Przedstawione tu rozwiązania możesz również stosować w znacznie bardziej rozbudowanych projektach, o ile tylko dobrze przyswoisz sobie podany przeze mnie tok postępowania. Plik z gotowymi wynikami ćwiczenia (*Acting on user input.fla*) oraz inne materiały dotyczące

ćwiczeń z niniejszej książki znajdziesz na serwerze FTP wydawnictwa „Helion", pod adresem: *ftp://ftp.helion.pl/przyklady/flmxas.zip*. Wykonaj następujące czynności:

1. Zacznij pracę w nowym, pustym pliku Flasha.
2. Włącz narzędzie *Text* i dodaj pole tekstowe w lewym górnym rogu sceny. Domyślnie będzie to pole statyczne (*Static Text*).
3. W polu tym wprowadź następujący tekst: *Please enter your name:* (lub po polsku: *Podaj swoje imię:* — w tym przypadku będziesz jednak musiał użyć jednej z czcionek z polskimi znakami diakrytycznymi).
4. Dodaj drugie puste pole tekstowe obok pierwszego, aby otrzymać układ podobny do tego na rysunku.



5. Po zaznaczeniu drugiego pola tekstowego otwórz panel *Properties* (polecenie *Window/Properties* lub skrót *Ctrl+F3*).
6. Z rozwijanej listy w lewym górnym rogu panelu *Properties* wybierz pozycję *Input Text*. Pozwoli to na wprowadzanie w nim tekstu podczas wyświetlania filmu.
7. W panelu *Properties* znajduje się pole oznaczone etykietą *Var*. W polu tym wprowadź nazwę `UserName` — tak będzie się nazywała zmienna odpowiadająca łańcuchowi znaków wprowadzonemu przez widza.
8. Upewnij się, że ikona *Render Text as HTML* nie jest włączona. Ponieważ będziemy za chwilę definiowali mechanizm sprawdzania poprawności wpisu, nie chcemy, aby program dołączył do tekstu znaczniki HTML, które tylko utrudniłyby nam to zadanie.
9. Kliknij ikonę *Show Border Around Text*. Dzięki temu puste pole zostanie otoczone ramką i użytkownik będzie wiedział, gdzie należy wprowadzić tekst. Panel *Properties* powinien teraz wyglądać następująco:



10. Otwórz bibliotekę z gotowymi zestawami przycisków, wybierając polecenie *Window/Common Libraries/Buttons*.
11. Przeciągnij klon jednego z przycisków do sceny i ulokuj go poniżej pól tekstowych. Możesz wybrać dowolny przycisk, który Ci się spodoba.
12. Utwórz kolejne pole typu *Static Text* na tle przycisku i wprowadź w nim napis *Continue* (lub po polsku: *Kontynuuj*). Flash pamięta typ ostatnio tworzonego pola tekstowego, dlatego konieczna będzie zmiana typu nowego pola z *Input Text* na *Static Text*. Rezultat powinien wyglądać podobnie jak na poniższym rysunku.



13. Kliknij klatkę 2. na listwie czasowej i wstaw pustą klatkę kluczową poleceniem *Insert/Blank Keyframe* lub klawiszem *F7*. Najlepiej jest użyć pustej klatki kluczowej, ponieważ nie chcemy, aby elementy z pierwszej klatki filmu były widoczne w drugiej klatce.

14. Dodaj pole tekstowe do sceny.

15. W panelu *Properties* zmień typ pola na *Dynamic Text*. Umożliwi nam to wyświetlanie w nim dowolnego tekstu utworzonego dynamicznie w kodzie ActionScript.

16. W polu *Var* wprowadź nazwę `Greeting` dla zmiennej reprezentującej zawartość pola tekstowego. Panel *Properties* powinien wyglądać jak na rysunku poniżej. W tej chwili możesz już zamknąć ten panel, ponieważ nie będzie nam więcej potrzebny.



Tworzenie skryptu

Wykorzystamy teraz kilka prostych linii kodu w języku ActionScript, które zmuszą film do działania zgodnego z naszymi założeniami. Wykonaj następujące czynności:

1. Zaznacz pierwszą klatkę na listwie czasowej filmu.

2. Otwórz panel *Actions*, używając polecenia *Window/Actions* lub skrótu klawiaturowego *F9*.

3. Wybierz podwójnym kliknięciem akcję `stop`, aby dodać ją do okna skryptu. Film powinien zatrzymać się w pierwszej klatce, co pozwoli użytkownikowi na wprowadzenie imienia do formularza i użycie przycisku.

4. Zaznacz przycisk umieszczony w scenie, wcześniej upewniając się, że żaden inny obiekt nie będzie zaznaczony wraz z przyciskiem.

5. W panelu *Actions* wybierz podwójnym kliknięciem akcję `goto` i w ten sposób dodaj ją do okna skryptu (zakładam, że pracujesz cały czas w trybie *Normal*).

6. Z rozwijanej listy *Type* wybierz pozycję *Next Frame*, dzięki czemu akcja przypisana przyciskowi będzie powodować przeskok do drugiej klatki filmu.

7. Zaznacz drugą klatkę na listwie czasowej filmu, gdyż teraz musimy umieścić w niej kolejny skrypt.

8. W panelu *Actions* wybierz podwójnym kliknięciem akcję `if` (z katalogu *Conditions/Loops*), co spowoduje jej wpisanie do okna skryptu.

9. W polu *Condition* wprowadź następujący warunek logiczny: `UserName + 1 == 1`. Będzie on sprawdzał, czy użytkownik wpisał jakikolwiek łańcuch znaków w polu formularza.



Mogłoby się wydawać, że prościej byłoby sprawdzić, czy łańcuch znaków w polu wejściowym ma postać "" lub " ", jednak ani w jednym, ani w drugim przypadku rezultat testu nie byłby prawidłowy. Jeśli zmienna we Flashu nie została w żaden sposób zainicjalizowana, przypisywana jest jej wartość pusta (`null`). Wartość pusta plus 1 daje w wyniku 1, dlatego wynik zdefiniowanego przez nas testu zawsze wykryje, czy pole było puste, czy też zawierało jakieś znaki. Gdy użytkownik wprowadzi choć jedną literę, wynikiem testu będzie wartość `false`. Odpowiednikiem powyższego testu mógłby być też następujący zapis: `UserName == null`.

10. W następnej linii kodu dodaj zapis `prevFrame();`, wybierając podwójnym kliknięciem akcję `goto` i wskazując pozycję *Previous Frame* na liście *Type*. W ten sposób film automatycznie powróci do pierwszej klatki, jeśli użytkownik nie wprowadzi tekstu do formularza.

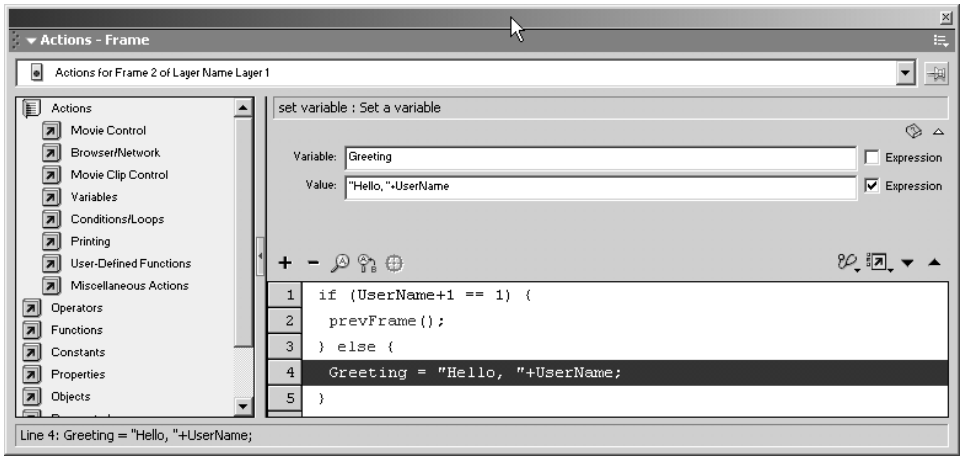
11. W trzeciej linii skryptu dodaj akcję `else`. Dzięki niej będziesz mógł zadecydować o tym, co ma nastąpić, gdy użytkownik wprowadzi swoje imię.

12. Jako następną akcję w skrypcie wybierz pozycję `set variable` z katalogu *Variables*.

13. W polu *Variable* wpisz słowo `Greeting` jako nazwę zmiennej. Upewnij się, że opcja *Expression* obok tego pola nie jest włączona.

14. Włącz opcję *Expression* obok pola *Value*, aby możliwe było wprowadzenie wyrażenia definiującego wartość zmiennej.

15. W polu *Value* wpisz wyrażenie `"Hello, " + UserName` (albo po polsku: `"Cześć, " + UserName`). Panel *Actions* powinien teraz wyglądać jak na poniższym rysunku.



Możesz przejść do testowania filmu. Zanim wprowadzisz swoje imię, spróbuj zobaczyć, co się stanie, jeśli użyjesz przycisku *Continue*. Powinieneś zauważyć, że w takiej sytuacji

odtwarzacz nie przechodzi do drugiej klatki filmu. Dopiero po wpisaniu jakiegokolwiek łańcucha znaków przycisk *Continue* przeniesie Cię do ekranu z komunikatem powitalnym i wyświetli tekst przypisany zmiennej `Greeting`.

Omówiony tu przykład jest oczywiście bardzo prosty i poza wyświetleniem powitalnego napisu nie realizuje żadnych innych zadań, jednak zademonstrował Ci działanie pewnych mechanizmów, które od teraz bez problemów będziesz mógł wykorzystywać w swoich filmach.

Wysyłanie danych do serwera

Dane wprowadzone przez użytkownika można nie tylko wykorzystać w filmie, ale także wysłać do serwera. Przesyłanie danych możliwe jest albo w specjalnym zbiorze utworzonym przez Flasha, albo w postaci e-maila. W obu przypadkach dane mogą zostać odebrane przez program nie związany z odtwarzaniem filmu Flasha.



Wadą przesyłania danych e-mailem jest to, że użytkownik musi potwierdzić wysłanie listu w programie do obsługi poczty elektronicznej. Zaletą jest natomiast to, że nie jest konieczne instalowanie żadnego specjalnego programu na serwerze odbierającym dane, co w wielu przypadkach jest bardzo istotne. Wysyłanie danych z formularza poprzez pocztę elektroniczną jest jednak niezbyt eleganckim rozwiązaniem z technicznego punktu widzenia.

Być może zdziwi Cię to, ale do wysyłania danych przez Internet wystarczy pojedyncza linia kodu w języku ActionScript. Mowa tu o akcji `getURL`, która pozwala wysyłać wartości wszystkich zmiennych filmu pod wskazany adres URL. Po wprowadzeniu akcji w panelu *Actions* możesz wybrać jedną z dwóch metod wysyłania danych: `GET` lub `POST`. Zazwyczaj do wysyłania niewielkich zbiorów informacji wykorzystujemy metodę `GET`, natomiast w przypadku większych ilości danych lepiej jest posłużyć się metodą `POST`.



Akcja `getURL` może wysłać dane tylko pod adres znajdujący się w tej samej subdomenie co film Flasha. Nie dotyczy to przesyłania danych pocztą elektroniczną — w tym przypadku informacje mogą być kierowane pod dowolny adres.

Pewnym utrudnieniem w przesyłaniu danych akcją `getURL` jest to, że nie można bezpośrednio wybrać, które informacje mają zostać wysłane, a które nie. Program domyślnie wysyła wartości wszystkich zmiennych, jednak możesz posłużyć się pewną sztuczką, jeśli chcesz ograniczyć wysyłany zbiór danych do wybranych pozycji. Zamiast umieszczać akcję `getURL` na głównej listwie czasowej, możesz umieścić ją w klonie klipu filmowego. Jeśli uprzednio przekażesz wartości odpowiednich zmiennych do tego klipu, tylko one zostaną przekazane do serwera.

Tworzenie własnego kursora

Czasami przydatne okazuje się stworzenie własnego kursora myszy, który w filmie Flasha zastąpi standardowo wyświetlany kursor. Na przykład, jeśli projektujesz grę, może okazać się przydatny kursor w kształcie paletki, którym gracz będzie mógł odbijać piłkę lub inne obiekty. Zastąpienie standardowego kursora we Flashu nie jest trudnym zadaniem, ponieważ

możesz ukryć go odpowiednią akcją ActionScript i zamiast niego wyświetlić odpowiednio przygotowany klip filmowy.



W danej chwili można przeciągać kursorem tylko jeden klip filmowy. Jeśli zastąpisz standardowy kursor myszy klipem filmowym, nie będziesz mógł bezpośrednio przemieszczać innego klipu akcją `startDrag`. Przy użyciu odpowiednich mechanizmów ActionScript można jednak rozwiązać ten problem. Wielu twórców pracujących z Flashem napisało własne wersje klasycznej już gry „Pong", stosując omawiane tu techniki.

W niniejszym przykładzie utworzymy kursor o prostych kształtach i pozwolimy widzowi przesuwać go myszą po ekranie. Później przyjrzymy się kilku rozwiązaniom, które mogą udoskonalić filmy zawierające własnoręcznie opracowane kursory.

Przygotowanie klipu filmowego

Aby podmienić standardowy kursor Flasha, musisz przygotować klip filmowy, który będzie odpowiednio reagował na ruch myszy. Zazwyczaj najprostszym sposobem realizacji tego zadania jest utworzenie symbolu typu *Movie Clip*, a następnie dołączenie do niego przycisku (symbolu typu *Button*). Jest to dość wygodne rozwiązanie, ponieważ z jednej strony — w przycisku zdefiniowane są uchwyty zdarzeń związanych z myszą, zaś z drugiej strony — klip filmowy można w łatwy sposób przesuwać kursorem. Łącząc te dwa typy symboli, otrzymasz obiekt, który z powodzeniem zastąpi kursor wyświetlany standardowo w filmach Flasha.

Plik z gotowymi wynikami ćwiczenia (*CUSTOM CURSOR.FLA*) znajdziesz na serwerze FTP wydawnictwa „Helion", pod adresem: *ftp://ftp.helion.pl/przyklady/flmxas.zip*.

Na początku utworzymy oba niezbędne symbole:

1. Otwórz nowy, pusty dokument Flasha.
2. Wybierz polecenie *Insert/New Symbol* lub naciśnij klawisze *Ctrl+F8*, aby wyświetlić okno dialogowe *Create New Symbol*.
3. Wprowadź dowolną nazwę dla nowego symbolu i włącz opcję *Button*, zanim klikniesz przycisk *OK*.
4. Narysuj figurę, której kształty ma posiadać nowy kursor.



Jeśli chcesz używać metody `hitTest` do sprawdzania, czy kursor natrafił na jakiś obiekt, najlepiej będzie posłużyć się kursorem w kształcie kwadratu lub prostokąta.

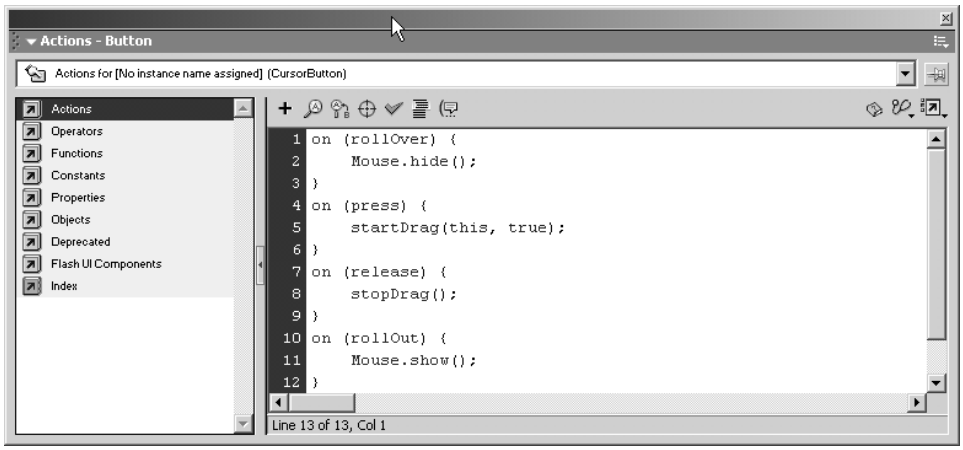
5. Otwórz panel *Align*, używając polecenia *Window/Panel/Align* lub skrótu klawiaturowego *Ctrl+K*.
6. Po zaznaczeniu narysowanego obiektu kliknij ikonę *To Stage*, a następnie użyj ikon *Align Horizontal Center* i *Align Vertical Center*, aby wyrównać obiekt do środka ekranu.
7. Ponownie wybierz polecenie *Insert/New Symbol* lub naciśnij *Ctrl+F8*, aby jeszcze raz otworzyć okno służącedo tworzenia nowego symbolu.

8. Wprowadź nazwę dla klipu filmowego i włącz opcję *Movie Clip* w polu *Behavior*, po czym kliknij przycisk *OK*.
9. Otwórz bibliotekę (*Library*) poleceniem *Window/Library* lub skrótem klawiaturowym *F11*.
10. Przeciągnij klon utworzonego przed chwilą przycisku z biblioteki do klipu filmowego.
11. Użyj narzędzi w panelu *Align* do ustawienia przycisku dokładnie pośrodku ekranu.

Dodawanie skryptu

Cały kod ActionScript umieścimy w obrębie przycisku, który przyłączymy do klipu filmowego. Ponieważ przycisk posiada już wbudowane mechanizmy związane z obsługą zdarzeń dotyczących kursora myszy, nie będziemy musieli zajmować się szczegółowo realizacją tego zadania.

Jak przedstawia kod na rysunku poniżej, uwzględnimy kilka różnych zdarzeń związanych z obsługą myszy. Omówię po kolei znaczenie każdego z nich.



Zaczniemy od ukrycia standardowego kursora myszy, gdy ten znajdzie się ponad klipem filmowym. Gdybyśmy tego nie zrobili, standardowy kursor byłby wyświetlany przez cały czas wraz z klipem filmowym. Kod użyty do wykonania tej operacji przedstawia się następująco:

```
on (rollOver) {
    Mouse.hide();
}
```

Po kliknięciu przycisku nowy kursor zostanie przyłączony do starego i będziesz mógł przesuwać go myszą dopóty, dopóki nie zwolnisz przycisku myszy. Realizuje to następujący kod:

```
on (press) {
    startDrag(this, true);
}
```

Gdy użytkownik zwolni przycisk myszy, klip filmowy zostanie zatrzymany w bieżącej pozycji. W tym celu należy wprowadzić taki oto zapis:

```
on (release) {
    stopDrag();
}
```

Jeśli po zwolnieniu przycisku użytkownik przesunie mysz poza klip filmowy, znów pojawi się standardowy kursor Flasha. Pozwoli na to następujący fragment kodu:

```
on (rollOut) {
    Mouse.show();
}
```

Aby dodać przedstawione wyżej bloki kodu do przycisku, wykonaj następujące czynności:

1. Włącz tryb edycji klipu filmowego i zaznacz znajdujący się w jego obrębie przycisk.
2. Otwórz panel *Actions*, jeśli nie jest widoczny na ekranie.
3. Z katalogu *Actions/Movie Control* wybierz podwójnym kliknięciem akcję `on`, aby dodać ją do okna skryptu.
4. W polu parametrów wybierz odpowiednie zdarzenie dla bloku kodu, który w danej chwili tworzysz.
5. W pierwszym i ostatnim bloku wybierz odpowiednią metodę związaną z obsługą myszy z katalogu *Objects/Movie/Mouse/Methods*.
6. W pozostałych dwóch blokach umieść, odpowiednio: akcje `startDrag` i `stopDrag`. W przypadku pierwszej z nich konieczne jest ustawienie parametrów tak jak w jednym z powyższych listingów. Akcja `stopDrag` nie wymaga podawania żadnych parametrów.

Przetestuj działanie filmu. Po kliknięciu klipu i przytrzymaniu lewego przycisku myszy standardowy kursor powinien zniknąć i podczas poruszania myszą po ekranie powinien przemieszczać się utworzony przez Ciebie kursor. Gdy zwolnisz przycisk myszy, standardowy kursor pojawi się z powrotem.

Dalsze możliwości

Zaprezentowany tu przykład z pewnością nie wyczerpuje możliwości związanych z tworzeniem własnych kursorów we Flashu, co oczywiście nie znaczy, że nie możesz rozbudować go we własnym zakresie. Oto kilka pomysłów, które być może Cię zainteresują:

- W obrębie klipu filmowego utwórz animację typu *Motion Tween*, w której kursor będzie powiększał się i pomniejszał pulsującym ruchem.
- Możesz użyć metody `attachMovie`, jeśli chcesz, aby klip pojawił się na ekranie w określonej chwili, a nie figurował tam przez cały czas, czekając na kliknięcie przez użytkownika.
- Jeśli chcesz napisać własną grę, możesz użyć metody `hitTest` do sprawdzenia, czy użytkownik trafił w piłkę, lotkę bądź inny obiekt (w zależności od tematu gry).

- Możesz utworzyć inny klip filmowy, który będzie „uciekał” po ekranie przed kursorem. Rezultat taki można osiągnąć na przykład poprzez przemieszczanie obiektu w losowym kierunku, gdy kursor znajdzie się w niewielkiej odległości od niego.
- Przy użyciu metody `hitTest` możesz wyświetlać serię obiektów podążających za kursorem w niewielkich odstępach. Niektórzy projektanci używają tej techniki do tworzenia ogona przypominającego kometę, który ciągnie się za przesuwanym przez użytkownika kursorem.