helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

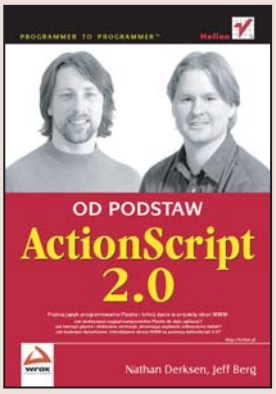
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

ActionScript 2.0. Od podstaw



Autorzy: Nathan Derksen, Jeff Berg
Tłumaczenie: Jarosław Dobrzański, Łukasz Schmidt
ISBN: 83-246-0655-6
Tytuł oryginału: Beginning ActionScript 2.0
Format: B5, stron: 872

Poznaj język programowania Flasha i tchnij życie w projekty stron WWW

- Jak dostosować wygląd komponentów Flasha do stylu aplikacji?
- Jak tworzyć płynne i efektowne animacje, zmieniając szybkość odtwarzania klatek?
- Jak budować dynamiczne, interaktywne strony WWW za pomocą ActionScript 2.0?

Jesteś użytkownikiem Flasha? Uważasz, że wiesz już wszystko o jego narzędziach graficznych i animacyjnych? A może Twoja wyobraźnia podsuwa Ci pomysły, których realizacja wydaje się niemożliwa? Dodaj do swojego warsztatu znajomość ActionScript 2.0, języka programowania wykorzystywanego we Flashu. Stosując go, nadasz nową jakość swoim projektom. Przekonasz się, że to, co było niemożliwe do wykonania za pomocą narzędzi rysunkowych, stanie się dziecinnie łatwe dzięki ActionScript 2.0. Będziesz mógł kontrolować niemal każdy parametr wszystkich obiektów w prezentacji, pobierać dane z zewnętrznych źródeł, sterować szybkością odtwarzania filmu i wykonywać wiele innych zadań.

„ActionScript 2.0. Od podstaw" to książka będąca wprowadzeniem do programowania w tym języku. Czytając ją, poznasz środowisko programistyczne Flasha i podstawy języka ActionScript. Dowiesz się, czym jest programowanie obiektowe i w jaki sposób korzystać z jego możliwości. Nauczysz się przy użyciu ActionScript kontrolować wartości parametrów klipów filmowych na scenie, sterować ich zachowaniem oraz tworzyć nowe obiekty. Przeczytasz o wstawianiu do prezentacji materiałów graficznych i dźwiękowych pochodzących z zewnętrznych źródeł, danych tekstowych, a także opanujesz sterowanie odtwarzaniem animacji. Znajdziesz tu również informacje o wyszukiwaniu i usuwaniu błędów w kodzie.

- Zadania panelu Actions
- Podstawowe elementy języka ActionScript
- Programowanie obiektowe
- Projektowanie aplikacji we Flashu
- Sterowanie klipami filmowymi
- Korzystanie z komponentów
- Tworzenie elementów interaktywnych
- Praca z tekstem
- Wstawianie do prezentacji elementów multimedialnych
- Dynamiczne tworzenie elementów graficznych
- Przetwarzanie plików XML
- Komunikacja z serwerem i przeglądarką

Pracujesz we Flashu? Wzbogać swoje projekty o możliwości, jakie daje Ci ActionScript

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

Spis tresci

O autorach 17

Wstęp 19

Rozdział 1. Rozpoczęcie pracy z programem Macromedia Flash 25

Wprowadzenie do środowiska autorskiego 25
Panel Tools 26
Panel Properties 26
Listwa czasowa 27
Klatki kluczowe i animacja 27
Biblioteka i symbole 28
Praca z przyciskami 29
Praca z klipami filmowymi 32
Warstwy, głębokości i poziomy 32
Dostosowywanie środowiska programistycznego 33
Panel Actions 33
Ustawienia ActionScript 34
Auto Format (automatyczne formatowanie kodu) 37
Publikacja projektu 38
Opcje wykrywania odtwarzacza Flash Player 42
Podsumowanie 43
Ćwiczenia 44

Rozdział 2. Rozpoczęcie pracy z ActionScript 2.0 45

Instrukcje 45
Stosowanie instrukcji prostych 45
Stosowanie instrukcji złożonych 46
Operatory 47
Priorytet operatorów 48
Porządek działania operatorów 49
Najpopularniejsze operatory 49

Białe znaki ... 51
Komentarze ... 52
Wstępne informacje o zmiennych ... 54
Przypisywanie danych do zmiennej ... 54
Odczytywanie zawartości zmiennej ... 55
Przekazywanie danych pomiędzy zmiennymi ... 55
Nazywanie zmiennych ... 56
Stałe ... 57
Ścisła kontrola typów a zmienne ... 57
Typy danych ... 60
Składnia kropkowa ... 64
Specjalne słowa kluczowe i zmienne ... 66
Praca ze zbiorami danych ... 68
Tablice ... 68
Tablice asocjacyjne i obiekty ... 74
Podsumowanie ... 79
Ćwiczenia ... 79

Rozdział 3. Konstrukcje warunkowe i pętle w ActionScript ... 81

Podejmowanie decyzji ... 81
Wyrażenia ... 82
Tworzenie wyrażeń ... 83
Instrukcja if..then..else ... 87
Instrukcja switch..case ... 89
Pętle ... 94
Pętla for ... 95
Pętla for..in ... 97
Pętla while ... 99
Pętla do..while ... 100
Wykrywanie błędów w kodzie pętli ... 104
Podsumowanie ... 108
Ćwiczenia ... 108

Rozdział 4. Funkcje i zakres w ActionScript ... 111

Funkcje ... 111
Funkcje w działaniu ... 114
Alternatywna składnia funkcji ... 116
Przekazywanie funkcji jako argument ... 116
Zakres zmiennych ... 120
Zarządzanie zakresem zmiennych ... 123
Skutki uboczne ... 125
Podsumowanie ... 127
Ćwiczenia ... 127

Rozdział 5. Rozpoczęcie tworzenia kodu ... 129

Wprowadzenie do programowania obiektowego ... 129
Definicja programowania obiektowego ... 129
Cele programowania obiektowego ... 134
Dobre praktyki w pisaniu kodu ... 135
Nadawanie nazw zmiennym ... 136
Kontrola typów ... 139

Komentowanie kodu ... 144
Formatowanie kodu ... 146
Zakres zmiennych ... 147
Dostęp do zmiennych innej listwy czasowej ... 150
Tworzenie własnych funkcji ... 157
Podsumowanie ... 169
Ćwiczenia ... 170

Rozdział 6. Przygotowanie projektów Flasha ... 173

Przygotowanie biblioteki ... 173
Praca z obrazami rastrowymi (bitmapami) ... 174
Organizowanie obrazów na pulpicie ... 175
Organizowanie elementów w bibliotece ... 175
Wbudowywanie obrazów w klipy filmowe ... 176
Zagnieżdżanie sekcji wewnątrz klipu filmowego ... 179
Zarządzanie stanem aplikacji za pomocą klatek kluczowych ... 180
Przechowywanie kodu w plikach zewnętrznych ... 183
Używanie skryptów do zarządzania stanem aplikacji ... 185
Podsumowanie ... 189
Ćwiczenie ... 189

Rozdział 7. Sterowanie klipami filmowymi za pomocą kodu ... 191

Natura klipów filmowych ... 191
Metody klasy MovieClip ... 192
Właściwości klasy MovieClip ... 206
Zdarzenia klasy MovieClip ... 209
Dynamiczne tworzenie klipów filmowych ... 210
Dołączenie klipu znajdującego się w bibliotece ... 214
Ładowanie filmów zewnętrznych ... 217
Ładowanie filmów do poziomów ... 217
Wczytywanie mediów do istniejących klipów filmowych ... 218
Adresy URL pełne, bezwzględne oraz względne ... 219
Stosowanie klipów filmowych jako masek ... 223
Zwiększanie wydajności klipów filmowych ... 228
Buforowanie bitmap ... 229
Przezroczystość ... 229
Filtry i tryby przenikania ... 230
Pełny ekran ... 230
Podsumowanie ... 230
Ćwiczenia ... 231

Rozdział 8. Tworzenie programów ładujących ... 233

Tworzenie własnego programu ładującego ... 233
Sprawdzanie postępu za pomocą zdarzenia onEnterFrame() ... 238
Klasa MovieClipLoader ... 239
Zdarzenia klasy MovieClipLoader ... 239
Metody klasy MovieClipLoader ... 240
Stosowanie klasy MovieClipLoader ... 243
Komponenty Leader oraz ProgressBar ... 246
Metody, właściwości oraz zdarzenia komponentu Loader ... 247
Metody, właściwości oraz zdarzenia komponentu ProgressBar ... 248
Stosowanie komponentów Loader oraz ProgressBar ... 249

Strategie stosowania programów ładujących ... 251
Film jako jednolita całość ... 252
Ładowanie filmu podzielonego na części ... 256
Podsumowanie ... 260
Ćwiczenia ... 260

Rozdział 9. Praca z komponentami ... 263

Nowości w wersji 2.0 ... 263
Przegląd komponentów ... 264
Grupa komponentów danych ... 265
Grupy komponentów FLV Playback oraz FLV Playback Custom UI ... 265
Grupa komponentów Media dla odtwarzaczy Flash Player 6 i 7 ... 267
Grupa komponentów User Interface ... 267
Ręczne umieszczanie komponentów na scenie ... 269
Umieszczanie komponentów na scenie za pomocą skryptu ... 272
Kontrolowanie komponentów za pomocą kodu ... 276
Podsumowanie ... 279
Ćwiczenia ... 280

Rozdział 10. Tworzenie interakcji z użytkownikiem ... 281

Obsługa zdarzeń ... 281
Tworzenie obiektu nasłuchującego ... 282
Tworzenie obiektu nasłuchującego — inna wersja ... 289
Tworzenie funkcji nasłuchującej ... 290
Której metody używać? ... 292
Dodawanie kilku obiektów nasłuchujących do kilku komponentów ... 297
Organizacja kodu odpowiedzialnego za wykrywanie zdarzeń ... 298
Obsługa zdarzeń pochodzących z wielu źródeł ... 299
Samodzielne wywoływanie zdarzeń dla komponentów ... 305
Podsumowanie ... 306
Ćwiczenie ... 307

Rozdział 11. Sterowanie komponentami ... 309

Panel Component Inspector ... 309
Tworzenie powiązań danych między komponentami za pomocą zakładki Bindings ... 311
Wykorzystanie pliku XML jako źródła danych ... 314
Sterowanie wyglądem komponentów ... 318
Zmiana stylów komponentów za pomocą metody setStyle() ... 318
Tworzenie skórek ... 326
Podsumowanie ... 331
Ćwiczenia ... 331

Rozdział 12. Wykrywanie i usuwanie błędów ... 333

Rodzaje błędów ... 333
Błędy kompilacji ... 333
Błędy logiczne ... 334
Projektowanie kodu ułatwiającego wykrycie błędów ... 340
Twórz kod łatwy do zrozumienia ... 340
Buduj kod po kawałku ... 341
Używaj małych funkcji ... 341

Naukowa metoda badawcza a wykrywanie błędów ... 341
Formułowanie teorii ... 342
Przeprowadzenie eksperymentu ... 343
Analiza otrzymanych wyników ... 351
Podsumowanie ... 352
Ćwiczenia ... 352

Rozdział 13. Praca z grafiką wektorową ... 357

Rysowanie za pomocą interfejsu API ... 358
Narzędzia służące do rysowania grafiki wektorowej w ActionScript ... 358
lineStyle() ... 359
beginFill() ... 360
beginBitmapFill() ... 360
beginGradientFill() ... 361
endFill() ... 363
moveTo() ... 363
lineTo() ... 363
curveTo() ... 363
clear() ... 364
Rysowanie wektorów za pomocą kodu ActionScript ... 364
Obiekt Matrix ... 373
Podsumowanie ... 374
Ćwiczenia ... 374

Rozdział 14. Tworzenie efektów specjalnych za pomocą filtrów ... 375

DropShadowFilter ... 376
BlurFilter ... 377
GlowFilter ... 378
BevelFilter ... 379
GradientGlowFilter ... 380
GradientBevelFilter ... 381
ConvolutionFilter ... 382
ColorMatrixFilter ... 385
DisplacementMapFilter ... 387
Klonowanie filtrów ... 388
Stosowanie wielu filtrów ... 389
Stosowanie trybów przenikania ... 394
Podsumowanie ... 395
Ćwiczenia ... 396

Rozdział 15. Modyfikowanie bitmap ... 397

Metody obiektu BitmapData ... 398
applyFilter() ... 398
clone() ... 398
colorTransform() ... 399
copyChannel() ... 399
copyPixels() ... 399
dispose() ... 400
draw() ... 400
fillRect() ... 401

floodFill() ... 401
generateFilterRect() ... 402
getColorBoundsRect() ... 402
getPixel() ... 403
getPixel32() ... 403
hitTest() ... 403
loadBitmap() ... 404
merge() ... 404
noise() ... 405
paletteMap() ... 405
perlinNoise() ... 406
pixelDissolve() ... 407
scroll() ... 408
setPixel32() ... 408
threshold() ... 409
Właściwości obiektu BitmapData ... 410
Konwertowanie klipu filmowego na bitmapę ... 410
Modyfikowanie obiektu BitmapData ... 411
Podsumowanie ... 413
Ćwiczenia ... 413

Rozdział 16. Wykorzystanie kodu ActionScript w animacjach ... 415

Rodzaje animacji we Flashu ... 416
Tworzenie animacji za pomocą klatek kluczowych ... 416
Tworzenie animacji za pomocą kodu ActionScript ... 418
Przesuwanie klipu filmowego za pomocą zdarzenia onEnterFrame() ... 419
Przesuwanie klipu filmowego za pomocą funkcji setInterval() ... 421
Animacje oparte na klatkach a animacje oparte na upływie czasu ... 423
Wpływ szybkości odtwarzania klatek na animację ... 423
Której metody używać? ... 426
Dobór szybkości odtwarzania lub parametru funkcji setInterval() ... 426
Animowanie klipów filmowych ... 427
Animowanie wielu klipów filmowych ... 428
Dodawanie zachowania losowego ... 429
Łagodzenie i przyspieszanie ... 434
Podsumowanie ... 448
Ćwiczenia ... 449

Rozdział 17. Automatyzacja animacji ... 451

Klasa Tween ... 451
Animacje względne i bezwzględne ... 452
Wbudowane klasy i metody upłynniające ruch ... 452
Metody klasy Tween ... 454
Właściwości i zdarzenia klasy Tween ... 459
Równoległe odtwarzanie sekwencji animacji ... 462
Tworzenie sekwencji płynnych przejść ... 466
Animowanie za pomocą kreślarskiego interfejsu programistycznego ... 473
Podsumowanie ... 481
Ćwiczenia ... 481

Rozdział 18. Praca z tekstem ... 483
Tworzenie pól tekstowych w ActionScript ... 484
Nazwy instancji pola a nazwy zmiennych pola ... 484
Czcionki systemowe i osadzone ... 484
Tworzenie pola tekstowego w locie ... 484
Operowanie właściwościami wyświetlania tekstu ... 486
Właściwość antiAliasType ... 487
Właściwość sharpness ... 487
Właściwość thickness ... 487
Możliwości w zakresie wzbogaconego formatowania tekstu ... 490
Korzystanie z klasy TextFormat ... 490
Nowe opcje klasy TextFormat we Flashu 8 ... 491
Słowo na temat metody setNewTextFormat() ... 497
Wyświetlanie tekstu HTML ... 497
Obsługa obrazów i plików SWF w polach tekstowych HTML ... 499
Obsługa czcionek ... 502
Używanie kaskadowych arkuszy stylów ... 504
Tworzenie obiektu kaskadowych arkuszy stylów ... 505
Przypisywanie stylu do pola tekstowego ... 505
Definiowanie stylów bezpośrednio w obiekcie StyleSheet ... 507
Jak definiować znaczniki, klasy i właściwości? ... 509
Stosowanie arkuszy CSS z XML ... 511
Opcje przewijania tekstu ... 513
Przewijanie tekstu za pomocą komponentu TextArea ... 514
Przewijanie tekstu za pomocą komponentu ScrollBar ... 515
Podsumowanie ... 522
Ćwiczenia ... 523

Rozdział 19. ActionScript i multimedia ... 525
Operowanie obrazami ... 525
Inteligentne ładowanie wstępne ... 527
Ładowanie wstępne za pomocą klasy MovieClipLoader ... 527
Operowanie dźwiękiem ... 529
Metody klasy Sound ... 529
Zdarzenia i właściwości klasy Sound ... 530
Tworzenie obiektu Sound ... 531
Ładowanie dźwięków z biblioteki ... 531
Ładowanie zewnętrznych plików MP3 ... 532
Odtwarzanie strumieniowe plików MP3 ... 534
Uruchamianie, zatrzymywanie i śledzenie obiektu dźwiękowego ... 534
Dźwięki zdarzeniowe ... 538
Ustawianie głośności, panoramy oraz używanie metody setTransform() ... 538
Praca z mikrofonem ... 545
Metody klasy Microphone ... 546
Właściwości i zdarzenia klasy Microphone ... 547
Aktywność mikrofonu ... 548
NetStream ... 550
Podsumowanie ... 551
Ćwiczenia ... 551

Rozdział 20. Praca z materiałami wideo ... 553
Pojęcia, technologie oraz jakość ... 553
Szybkość transmisji danych ... 554
Pobieranie progresywne — HTTP ... 554
Strumieniowanie — Flash Communication Server ... 554
Klatki kluczowe materiału wideo ... 555
Zmienna szybkość transmisji bitów ... 556
Przeplot ... 556
Szybkość odtwarzania klatek ... 556
Tworzenie materiału w formacie Flash Video ... 557
Konwertowanie materiałów wideo ... 557
Ładowanie zewnętrznej zawartości wideo ... 561
Eksportowanie wideo ... 561
Ładowanie zewnętrznego pliku FLV ... 562
Klasa NetStream ... 562
Klasa Video ... 564
Kontrolowanie punktu odtwarzania wideo ... 566
Obsługa przezroczystości ... 569
Poprawa jakości materiałów wideo ... 572
Redukcja bloków i obwódek ... 572
Skalowanie i przezroczystość ... 574
Praca z kamerą ... 574
Metody, właściwości i zdarzenia klasy Camera ... 575
Tworzenie obiektu typu Camera ... 576
Wyświetlanie materiału wyjściowego z kamery na scenie ... 577
Zagadnienia bezpieczeństwa związane z wyskakującymi powiadomieniami ... 582
Podsumowanie ... 583
Ćwiczenia ... 583

Rozdział 21. Komunikacja z serwerem za pomocą kodu ActionScript ... 585
Dane zewnętrzne a użyteczność aplikacji ... 586
Opóźnienia transmisji danych: komunikacja asynchroniczna ... 586
Klasa LoadVars ... 587
Dane rozdzielone znakami & ... 587
Metody klasy LoadVars ... 588
Tworzenie obiektu LoadVars ... 590
Procedury obsługi zdarzeń klasy LoadVars ... 592
Ładowanie i przetwarzanie tekstu ... 595
Podsumowanie ... 598
Ćwiczenia ... 598

Rozdział 22. Wczytywanie dokumentów XML ... 599
Język XML ... 599
Znaczniki ... 599
Sekcje CDATA ... 601
Klasa XML ... 602
Metody i właściwości klasy XML ... 602
Zdarzenia klasy XML ... 606
Właściwość ignoreWhite ... 607

Ładowanie zewnętrznych plików XML ... 607
Sprawdzanie postępu ładowania pliku ... 610
Bezpieczeństwo plików lokalnych i zewnętrznych ... 610
Specyfika działania metody getBytesTotal() ... 614
Relacje pomiędzy węzłami rodzicami a węzłami potomnymi ... 614
Nawigacja po przykładowym drzewie XML ... 615
Korzystanie z atrybutów ... 618
Połączenia za pomocą XML Socket ... 618
XML w połączeniu z klasą XMLSocket ... 619
Puste bajty ... 619
Metody i zdarzenia klasy XMLSocket ... 620
Metoda sendAndLoad() ... 622
Metody GET i POST protokołu HTTP ... 624
Różnice w działaniu metod HTTP GET i POST ... 625
Kryteria wyboru metody ... 626
Praca z komponentem XMLConnector ... 626
Poruszanie się między domenami a zabezpieczenia ... 630
Aplikacje Flasha a zasady bezpieczeństwa domen ... 631
Kiedy zastosowanie plików zasad będzie konieczne? ... 632
Tworzenie plików zasad ... 633
Usługi sieciowe i dostęp z dowolnej domeny ... 635
Tworzenie podkładek i skryptów pośredniczących ... 638
Podsumowanie ... 642
Ćwiczenia ... 643

Rozdział 23. Komunikacja pomiędzy nakładką Macromedia Flash a przeglądarką ... 645

Połączenia lokalne – klasa LocalConnection ... 645
Tworzenie obiektu LocalConnection ... 647
Bezpieczeństwo ... 649
Przechowywanie danych lokalnie za pomocą obiektów współdzielonych ... 650
Akceptowalne typy danych ... 652
Wykorzystanie obiektów SharedObject jako plików cookie ... 653
Współpraca z użytkownikiem ... 656
Zarządzanie przestrzenią dyskową ... 656
Dzielenie obiektów współdzielonych ... 657
Przekazywanie danych wejściowych za pomocą FlashVars ... 658
Wprowadzenie do parametrów znaczników <object> i <embed> ... 658
Dodawanie zmiennych FlashVars ... 660
Tworzenie zmiennych FlashVars za pomocą kodu JavaScript ... 661
Przekazywanie FlashVars za pośrednictwem strony serwletowej ... 662
Podsumowanie ... 663
Ćwiczenia ... 664

Rozdział 24. Praca z JavaScript ... 665

Zmiana ustawień zabezpieczeń ... 665
Wywoływanie kodu JavaScript z filmu Flasha ... 667
Wprowadzenie do metody getURL() ... 667
Tworzenie polecenia JavaScript za pomocą zmiennych zagnieżdżonych ... 668
Ograniczenia ... 672

Wywoływanie funkcji JavaScript za pomocą fscommand() ... 673
Wywoływanie kodu ActionScript z kodu JavaScript ... 675
Podstawowe metody i zdarzenia JavaScript ... 677
Pakiet Flash JavaScript Integration Kit ... 678
Przygotowanie pakietu Flash JavaScript Integration Kit ... 678
Wywoływanie funkcji JavaScript z kodu ActionScript ... 679
Wywoływanie funkcji ActionScript z kodu JavaScript ... 679
Zewnętrzny interfejs API ... 683
Metody klasy ExternalInterface ... 683
Wywoływanie funkcji i metod ActionScript ... 686
Wywoływanie funkcji JavaScript z poziomu kodu ActionScript ... 689
Otwieranie okien przeglądarki ... 700
Otwieranie wyskakujących okien za pomocą getURL() ... 700
Tworzenie wyskakujących okien za pomocą klasy ExternalInterface ... 701
Tworzenie funkcji JavaScript jako osłony ... 701
Definiowanie parametrów okna przeglądarki ... 702
Podsumowanie ... 704
Ćwiczenie ... 704

Rozdział 25. Przesyłanie i pobieranie plików ... 705

Metody klasy FileReference ... 705
browse() ... 706
cancel() ... 706
download() ... 707
upload() ... 707
Właściwości i zdarzenia klasy FileReference ... 708
Pobieranie plików ... 709
Przesyłanie plików ... 713
Podsumowanie ... 719
Ćwiczenie ... 719

Rozdział 26. Komunikacja pomiędzy nakładką a systemem operacyjnym ... 721

Odczytywanie informacji systemowych za pomocą obiektu System ... 722
Ograniczenia obiektu System w różnych wersjach odtwarzacza Flash Player ... 723
Metoda setClipBoard() ... 724
Projektory, pliki wykonywalne Flasha i inne pliki wykonywalne ... 724
Ograniczenia funkcji fscommand ... 725
Wywoływanie funkcji projektora za pomocą funkcji fscommand ... 725
Pakiet SDK ... 727
Oprogramowanie autorstwa firm niezależnych ... 727
Podsumowanie ... 728
Ćwiczenia ... 728

Rozdział 27. Tworzenie klas własnych ... 729

Praca z klasami ... 729
Definiowanie klasy ... 729
Słowa kluczowe public i private ... 732
Definiowanie konstruktora ... 734
Definiowanie metod ... 738

Definiowanie właściwości ... 744
Dodawanie funkcji do klasy istniejącej ... 751
Podsumowanie ... 764
Ćwiczenia ... 765

Dodatek A Rozwiązania ćwiczeń ... 769

Dodatek B Skróty klawiszowe ... 831

Skorowidz ... 843

1

Rozpoczęcie pracy z programem Macromedia Flash

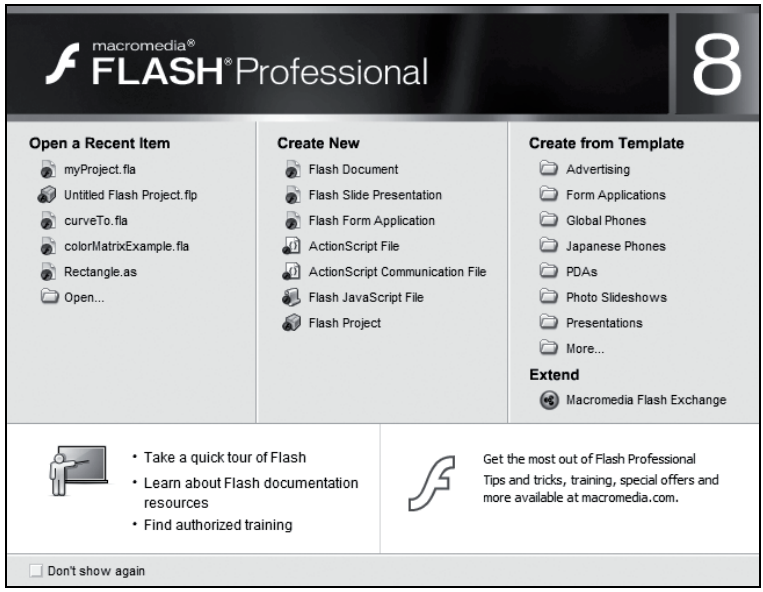
Pojęcie Flash oznacza zarówno format animacji opartych na grafice wektorowej (filmy Flasha), jak również narzędzie autorskie służące do ich tworzenia. Odtwarzacz Flash Player jest programem służącym do odtwarzania filmów Flasha. ActionScript to język skryptowy, którego kod dołączany do zawartości utworzonej za pomocą Flasha jest interpretowany przez odtwarzacz Flash Player. Pierwszym etapem podczas nauki języka ActionScript będzie opanowanie umiejętności związanych z wykorzystaniem narzędzia autorskiego, jakim jest Flash. W tym rozdziale dowiesz się, jak pracować z Flashem, a to da Ci podstawy niezbędne do pisania kodu ActionScript.

Wprowadzenie do środowiska autorskiego

W celu zorganizowania funkcji dostępnych w programie Flash korzysta z systemu paneli. Panele mogą być dokowane lub oddokowane („pływające”), a także przesuwane poza obszar aplikacji. Wszystko to pozwala na dostosowanie obszaru roboczego do własnych potrzeb. Opcje paneli i układu znajdziesz w menu *Window* (okno) znajdującym się w menu głównym. W wielu miejscach niniejszego rozdziału przyjęliśmy, iż pracujesz w domyślnym układzie paneli. Jeżeli np. jest mowa o tym, iż panel pojawia się w lewym górnym rogu obszaru roboczego, oznacza to, że dzieje się tak przy domyślnym układzie paneli. W każdej chwili możesz przywrócić domyślny układ paneli, wybierając *Window/Workspace Layout/Default*.

Po uruchomieniu przy domyślnych ustawieniach Flash 8 wyświetla stronę startową (rysunek 1.1), która pozwala na szybkie otwieranie projektów. Możesz określić, czy chcesz, aby program wyświetlał stronę startową. Jeżeli usuniesz zaznaczenie z pola *Don’t show again* (nie pokazuj ponownie), strona nie będzie wyświetlana. Ustawienia dotyczące strony startowej możesz zmienić także później za pomocą okna dialogowego *Preferences* (ustawienia); aby je wyświetlić, wybierz *Edit/Preferences* i wskaż odpowiednią opcję z menu *On launch*.

Rysunek 1.1.



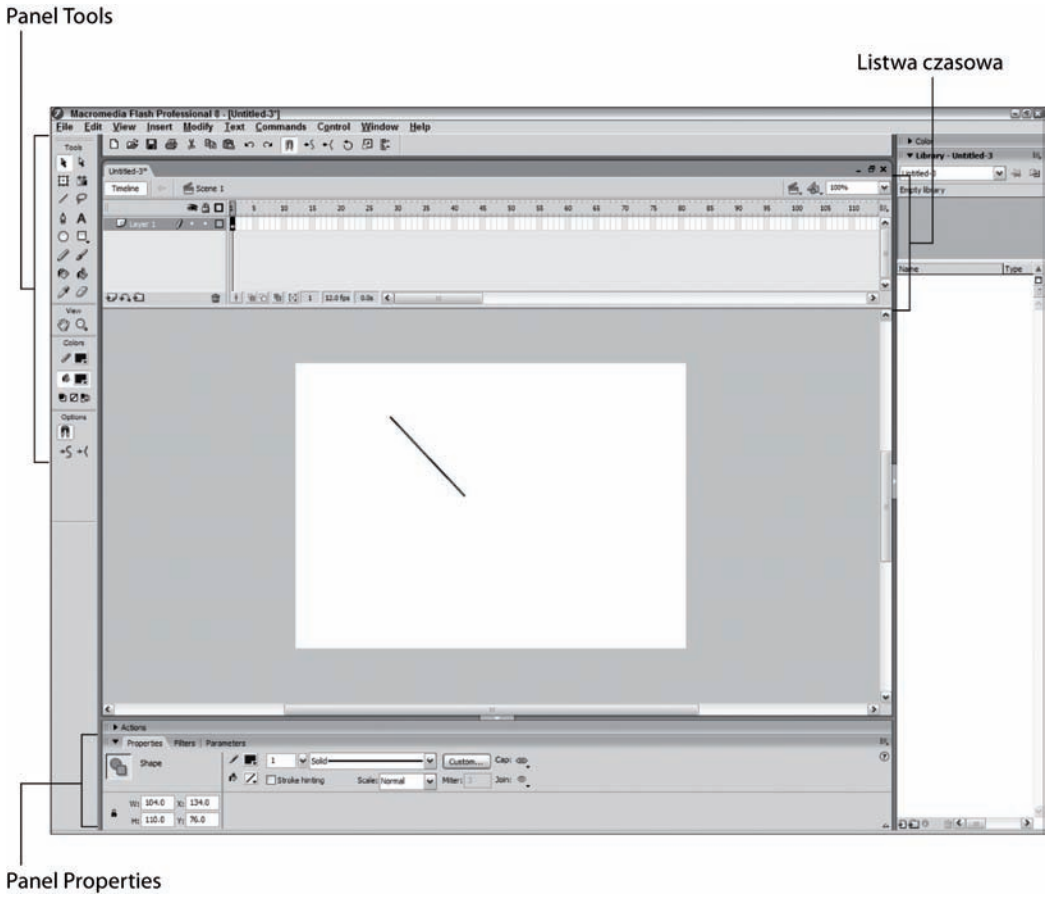
Poniżej przedstawimy pokrótce inne części środowiska autorskiego Flash, często określanego jako **IDE** (skrót od *Integrated Development Environment* — zintegrowane środowisko programistyczne).

Panel Tools

Po lewej stronie IDE znajduje się panel *Tools* (rysunek 1.2) zawierający narzędzia służące do rysowania nowej grafiki wektorowej i edycji grafiki już istniejącej. Tutaj będziesz wybierał narzędzia służce do rysowania, ustawiał ich atrybuty i opcje. Atrybuty narzędzi często są wyświetlane w panelu *Properties*, nazywanym także **inspektorem właściwości**. Panel ten znajduje się w dolnej części środowiska IDE. Jeżeli nie jest widoczny, wybierz *Window/Properties/Properties*.

Panel Properties

W panelu *Properties* (również widoczny na rysunku 1.2) znaleźć można właściwości obiektu zaznaczonego na scenie lub wybranego narzędzia. Ma **charakter kontekstowy** — jego zawartość jest różna, a zależy od zaznaczonego obiektu lub wybranego narzędzia. Jeżeli np. nie zaznaczyłeś niczego na scenie i wybrałeś narzędzie *Selection* (wybór), panel *Properties* wyświetli opcje pozwalające na dostosowanie ogólnych właściwości dokumentu, takich jak wymiary sceny i liczba klatek odtwarzanych w jednej sekundzie. Jeśli jednak wybierzesz narzędzie *Line* (linia), panel *Properties* pokaże jego właściwości, takie jak grubość i styl linii. Gdy natomiast za pomocą narzędzia *Selection* wybierzesz linię już umieszczoną na scenie, zobaczysz ten sam zestaw właściwości. Tym razem jednak, ponieważ zmieniasz właściwość zaznaczonej linii, znajdzie to natychmiast odzwierciedlenie w wyglądzie obiektu umieszczonego na scenie.



Rysunek 1.2.

Listwa czasowa

Listwa czasowa Flasha (rysunek 1.2) znajduje się powyżej głównego obszaru nawigacji. Listwa czasowa pozwala na dodawanie zawartości, która będzie zmieniać się wraz z upływem czasu. Istnienie listwy czasowej jasno wskazuje na to, iż Flash jest narzędziem służącym do tworzenia animacji.

Klatki kluczowe i animacja

Listwy czasowe pozwalają na dodawanie **klatek** (*frame*) i animację zawartości. Domyślnie listwa czasowa zawiera tylko jedną klatkę — klatkę 1. Możesz dodawać klatki, wybierając *Insert/Timeline/Frame* lub naciskając klawisz *F5*. Miejsce, w którym zostanie dodana nowa klatka, zależy od tego, która klatka jest aktualnie wybrana na listwie czasowej. Jeżeli zaznaczysz klatkę już istniejącą na listwie czasowej, Flash wstawi nową klatkę tuż za zaznaczoną i przesunie kolejne klatki w prawo o jedno miejsce. Jeżeli natomiast jako miejsce

wstawienia klatki wskażesz punkt za istniejącymi klatkami, Flash wstawi klatkę dokładnie w tym miejscu. Możliwe jest także, jeżeli będzie to konieczne, dodanie wszystkich klatek pomiędzy klatką nowo wstawioną a już istniejącymi. Gdy przykładowo listwa czasowa liczy tylko jedną klatkę, a Ty umieścisz punkt wstawiania klatki w miejscu klatki 5., Flash doda cztery nowe klatki; jedną w punkcie klatki 5. oraz trzy pomiędzy istniejącą klatką 1. a nową.

Flash korzysta także z **klatek kluczowych** (*keyframe*). Są to specjalne klatki, do których możesz dodawać zawartość. Zwykłe klatki służą tylko do określenia, ile czasu upłynie między klatkami kluczowymi. Grafikę, tekst i inne elementy możesz dodawać wyłącznie do klatek kluczowych. Pierwsza domyślna klatka na listwie czasowej jest zawsze klatką kluczową. Nowe klatki kluczowe dodasz, wybierając *Insert/Timeline/Keyframe* lub naciskając klawisz *F6*. Kiedy zaznaczasz istniejącą klatkę kluczową i dodasz nową klatkę kluczową, nowa będzie umieszczona bezpośrednio za zaznaczoną klatką. Jeżeli następująca po niej klatka jest klatką zwykłą, zostanie zamieniona na klatkę kluczową. Jeżeli już jest klatką kluczową, nie ma zmian. Jeśli po zaznaczonej klatce nie ma już dalszych klatek, po prostu dodawana jest nowa klatka kluczowa. Gdy ustawisz punkt wstawiania klatki w miejscu, gdzie nie ma jeszcze żadnej klatki, Flash doda nową klatkę kluczową oraz zwykłe klatki dla wszystkich klatek pomiędzy ostatnią istniejącą klatką, a nową klatką kluczową. Jeżeli zaznaczysz istniejącą klatkę i dodasz klatkę kluczową, Flash po prostu skonwertuje klatkę na klatkę kluczową.

Na listwie czasowej klatki kluczowe oznaczane są okrągłymi kropkami. Jeżeli klatka nie posiada zawartości, odpowiadająca jej kropka jest pusta w środku. Kiedy zawartość zostanie dodana, klatka oznaczona będzie czarną, wypełnioną kropką. Zwykłe klatki następujące za pustą klatką kluczową zaznaczone są kolorem białym, kiedy zaś poprzedzająca je klatka kluczowa ma zawartość — kolorem szarym.

Na poziomie dokumentu zawsze istnieje przynajmniej jedna listwa czasowa. Ta domyślna listwa czasowa najwyższego poziomu nazywana jest **główną listwą czasową** (*main timeline*) lub źródłową listwą czasową (*root timeline*). W ActionScript główna listwa czasowa ma dwie nazwy, czyli `_root` i `_level0`. Wiele rodzajów symboli umieszczonych w bibliotece (takich jak np. klipy filmowe) ma własne listwy czasowe. Częstym błędem popełnianym zarówno przez początkujących, jak i ekspertów Flasha jest pomyłkowe edytowanie niewłaściwej listwy czasowej. Możesz zorientować się, którą listwę czasową aktualnie edytujesz, patrząc na część panelu znajdującą się po prawej stronie przycisku *Timeline* (służącego do pokazywania i ukrywania listwy czasowej). Flash pokazuje nazwę aktualnie edytowanej listwy. Domyślnie Flash nadaje nazwę *Scene 1* głównej listwie czasowej. Jeżeli edytujesz symbol, jego nazwa pojawia się po prawej stronie nazwy *Scene 1*. W danym momencie edytujesz listwę czasową, która znajduje się najdalej, po prawej stronie od *Scene 1*. Zawsze możesz powrócić do głównej listwy czasowej, klikając nazwę *Scene 1*.

Biblioteka i symbole

Zawartość filmu Flasha składa się z elementów, takich jak tekst, grafika wektorowa, bitmapy, dźwięki i filmy. Aby efektywnie zarządzać zawartością, umieszcza się ją wewnątrz symboli. Flash zawiera wiele rodzajów symboli, a wśród nich przyciski, symbole graficzne, klipy filmowe, bitmapy, czcionki, dźwięki oraz filmy. Trzy rodzaje (przyciski, symbole graficzne

i klipy filmowe) pozwalają na grupowanie zawartości sceny i łączenie jej w celu ponownego wykorzystania. Symbole przechowywane są w bibliotece (*library*), która domyślnie wyświetlana jest po prawej stronie środowiska IDE. Możesz pokazywać i ukrywać jej panel, wybierając *Window/Library* (okno/biblioteka).

Nowy przycisk, symbol graficzny lub klip filmowy tworzymy na kilka sposobów. Możesz narysować go na scenie, zaznaczyć i skonwertować na symbol, wybierając *Modify/Convert to Symbol* lub naciskając klawisz *F8*. Możesz także wybrać *Insert/New Symbol* lub nacisnąć klawisze *Ctrl+F8* (*Cmd+F8* w Mac OS). Innym sposobem jest także naciśnięcie przycisku *New Symbol* w panelu *Library* i dodanie nowego, pustego symbolu. Za każdym razem otworzy się okno dialogowe zawierające prośbę o podanie nazwy i rodzaju nowego symbolu.

To, jaką nazwę nadasz symbolowi, zależy tylko od Ciebie — nie ma żadnych reguł, których musisz przestrzegać. Nazwa, której użyjesz, będzie także pokazywana obok symbolu w bibliotece. Warto nadawać nazwy odzwierciedlające zawartość symbolu. Jeżeli np. symbol zawiera rysunek chmury, możesz nadać mu nazwę `chmura`.

Ważne jest, jaki **typ symbolu** tworzysz. Dostępne są trzy opcje: *Movie clip* (klip filmowy), *Button* (przycisk) oraz *Graphic* (symbol graficzny). Każdy rodzaj symbolu zachowuje się inaczej i każdy powinien być używany do innych celów. Symbole graficzne przydatne są do łączenia razem grafik i (lub) sekwencji animowanych. Ponieważ za pomocą języka ActionScript nie możesz kontrolować symboli graficznych (ani ich instancji), nie będą one omawiane w tej książce. Symbole przycisków korzystają ze specjalnej listwy czasowej, złożonej z czterech klatek kluczowych oznaczonych *Up*, *Over*, *Down* oraz *Hit*. Kiedy instancję symbolu przycisku umieścisz na scenie, a następnie wyeksportujesz film, Flash będzie przechodził do klatek kluczowych automatycznie, w zależności od działań użytkownika. Kiedy np. użytkownik przesunie wskaźnik myszy nad instancję przycisku, Flash przejdzie do klatki *Over*. Więcej na temat symboli przycisków dowiesz się z następnego punktu, „Praca z przyciskami". Symbole przycisków są przydatne do tworzenia obiektów reagujących na kliknięcia. Największe jednak znaczenie, z punktu widzenia języka ActionScript, mają symbole klipów filmowych. Za pomocą ActionScript możesz dodawać instancje klipów filmowych i kontrolować je. Więcej na temat symboli klipów filmowych dowiesz się z punktu „Praca z klipami filmowymi", w dalszej części rozdziału.

Po tym, jak już dodasz do biblioteki symbole graficzne, przyciski lub klipy filmowe, będziesz dysponował zbiorem szablonów gotowym do ponownego wykorzystania. Możesz porównać go do zbioru planów poszczególnych obiektów. Jeżeli będzie to potrzebne, będziesz mógł dodać jedną lub więcej instancji danego obiektu na scenę. Zrobisz to, przeciągając symbol z biblioteki na scenę. Każda instancja będzie oparta na tym samym szablonie — symbolu w bibliotece. Jednak każda instancja będzie unikalna i niezależna od innych instancji tego samego symbolu. Usunięcie symbolu z biblioteki spowoduje usunięcie ze sceny wszystkich jego instancji.

Praca z przyciskami

Przyciski to symbole posiadające cztery stany: *Up*, *Over*, *Down* oraz *Hit*. Listwę czasową przycisku łatwo rozpoznać, ponieważ cztery klatki kluczowe opisane są odpowiednio do stanów. Do tych klatek, tak jak do klatek kluczowych na innych listwach czasowych, możesz dodawać instancje klipów filmowych, symboli graficznych, kod ActionScript, dźwięki i inne elementy.

Domyślnie listwy czasowe przycisków mają zdefiniowaną tylko jedną klatkę — stan *Up*. Jeżeli dodasz zawartość do stanu *Up*, następnie umieścisz instancję przycisku na scenie i przetestujesz film, zobaczysz, że wskaźnik myszy zmienia wygląd, kiedy przesuwasz go ponad instancję przycisku. Jeżeli zdefiniujesz klatkę kluczową dla klatki *Over* i dodasz do niej zawartość inną niż dla stanu *Up*, testując film, zobaczysz, iż po najechaniu wskaźnikiem myszy nad instancję przycisku jej zawartość zmieni się odpowiednio do stanu *Over*. Kiedy usuniesz wskaźnik myszy znad przycisku, wygląd powróci do stanu *Up*. Podobnie dodanie klatki kluczowej i zawartości do stanu *Down* spowoduje, iż Flash automatycznie przejdzie do tej klatki, kiedy użytkownik kliknie przycisk. Stan *Hit* nie będzie nigdy widoczny w czasie testowania filmu, definiuje on obszar aktywny, który opowiada za przywołanie stanów *Over* i *Down*. Jeżeli np. zawartość stanu *Hit* składa się wyłącznie z tekstu, użytkownik będzie musiał przesunąć wskaźnik myszy dokładnie nad krawędź czcionki, aby wywołać stany *Over* i *Down*. Jeśli wskaźnik nie będzie dokładnie nad literami, pozostałe stany nie zostaną wywołane. Jeśli jednak dla stanu *Hit* zdefiniujesz większy obszar, przesunięcie wskaźnika myszy w jego dowolne miejsce spowoduje uaktywnienie przycisku.

spróbuj sam Praca z symbolami przycisków

W tym ćwiczeniu utworzymy prosty przycisk interaktywny. Oto, co należy zrobić:

1. Utwórz nowy dokument Flasha. Wybierz *File/New*, a następnie w panelu *New Document* wybierz *Flash Document*. Zapisz go jako `symbolExample.fla`.
2. Jeżeli biblioteka nie jest widoczna, wybierz z menu głównego *Window/Library*.
3. Wybierz *Insert/New Symbol*.
4. Otworzy się okno dialogowe *Create New Symbol*. Nazwij nowy symbol `myButton`, upewnij się, że zaznaczona jest opcja *Button*. Nie zwracaj uwagi na pozostałe opcje. Kliknij *OK*.
5. Poszczególne stany przycisku i jego listwa czasowa widoczne są w górnej części IDE. Zaznacz pierwszą klatkę nazwaną *Up*.
6. Z panelu *Tools* znajdującego się w lewej części okna IDE wybierz narzędzie *Rectangle* (prostokąt).
7. W panelu *Tools* widoczne są dwie próbki kolorów. Górna to kolor linii, dolna — wypełnienia. Kliknij pole górnej próbki i wybierz kolor ciemnoszary lub czarny jako kolor linii. Kliknij próbkę koloru wypełnienia i wybierz kolor ciemnoczerwony.
8. Narysuj na scenie prostokąt. Kliknij i przeciągnij aż do uzyskania odpowiedniego kształtu. Nie przejmuj się ustawianiem prostokąta dokładnie wewnątrz symbolu. Prostokąt powinien wyglądać podobnie jak na rysunku 1.3.
9. Wybierz z panelu *Tools* narzędzie *Selection* (wygląda jak mała czarna strzałka). Zaznacz prostokąt, klikając dwukrotnie wewnątrz jego wypełnienia.
10. Panel *Properties* na dole IDE pokazuje teraz właściwości zaznaczonego prostokąta. Zmień jego wysokość (pole *H:*) na `30`, a szerokość na `120` (pole *W:*), ustawienie na osi x i y (pola *X:* i *Y:*) zmień na `0`.

Rysunek 1.3.

11. Zaznacz na listwie czasowej klatki *Over*, *Down* oraz *Hit*, kliknij pustą klatkę *Hit* i, trzymając wciśnięty przycisk myszy, przeciągnij wskaźnik w lewo aż do zaznaczenia wszystkich trzech klatek.

12. Mając ciągle zaznaczone wszystkie trzy klatki, kliknij prawym przyciskiem myszy (*option*+kliknięcie w Mac OS) klatkę *Over*. Z menu kontekstowego wybierz *Convert to Keyframes* (konwertuj na klatki kluczowe). Nowe klatki kluczowe będą zawierać kopie zawartości z klatki pierwszej, czyli kopię narysowanego prostokąta.

13. Zaznacz klatkę *Over*. Zaznacz prostokąt, klikając go za pomocą narzędzia *Selection*. Zmień kolor wypełnienia na różowy.

14. Zaznacz klatkę *Down*. Zaznacz prostokąt, klikając go za pomocą narzędzia *Selection*. Zmień kolor wypełnienia na biały.

15. Utwórz nową warstwę listwy czasowej, klikając ikonę *Insert Layer* (wstaw warstwę) w lewym dolnym rogu panelu *Timeline* (ikona wygląda jak kartka papieru ze znakiem plus (+)).

16. Zaznacz pierwszą klatkę na warstwie *layer2* i wybierz narzędzie *Text* (oznaczone dużą literą A) z panelu *Tools*.

17. Kliknij w dowolnym punkcie sceny, aby utworzyć pole tekstowe. W polu wpisz `Play`.

18. Wybierz z panelu *Tools* narzędzie *Selection* i kliknij tekst, który właśnie wpisałeś (jeżeli nie widzisz, gdzie znajduje się pole tekstowe, ponieważ jego kolor jest identyczny z kolorem sceny, kliknij ponownie pierwszą klatkę warstwy *layer2*). Panel *Properties* pokazuje teraz właściwości pola tekstowego. Kliknij próbkę koloru i zmień go na jasnoszary.

19. Wyrównaj tekst do środka prostokąta.

20. W panelu *Timeline* kliknij przycisk *Scene 1*. Przycisk zniknął ze sceny, gdyż znajduje się w bibliotece (panel *Library*). Otwórz bibliotekę, zaznacz przycisk i przeciągnij go na scenę, aby utworzyć jego instancję.

21. Przetestuj aplikację, wybierając *Control/Test Movie*.

22. Przetestuj przycisk, najeżdżając na niego wskaźnikiem myszy i klikając.

Jak to działa?

W tym ćwiczeniu skorzystałeś z narzędzi tworzenia grafiki wektorowej, biblioteki oraz ikon i opcji w panelu *Timeline* w celu szybkiego utworzenia interaktywnego przycisku reagującego na działania myszy. Zmieniłeś jego rozmiar i położenie.

W dalszej części książki dowiesz się, jak importować pliki obrazów do symboli przycisków dla każdego ze stanów. Za pomocą programów graficznych możesz w formatach *.jpeg*, *.gif* oraz *.png* tworzyć pliki obrazów, które posłużą do nadania przyciskowi ciekawszego wyglądu.

Zapisz dokument Flasha, wykorzystasz go w następnym ćwiczeniu z serii „Spróbuj sam".

Praca z klipami filmowymi

Symbole klipów filmowych posiadają własne listwy czasowe, które mogą być odtwarzane niezależnie od jakiejkolwiek listwy czasowej filmu Flasha. Możesz programowo kontrolować instancje symboli klipów filmowych, nadając im wyrafinowane funkcje. Te cechy sprawiają, że symbole klipów filmowych są najpopularniejszym rodzajem symboli wykorzystywanym w połączeniu z ActionScript. Klipy filmowe są tak ważne, że poświęcimy ich omówieniu osobny rozdział — rozdział 7.

Warstwy, głębokości i poziomy

Zawartość listew czasowych możesz **dzielić na warstwy** (*layer*). Jeżeli umieścisz zawartość w odrębnych warstwach, będziesz mógł zmieniać kolejność ułożenia warstw i tym samym określać, które obiekty zostaną umieszczone na wierzchu innych. Kolejność warstw zmieniasz, przeciągając je wewnątrz listwy czasowej.

Warstwy są wykorzystywane wyłącznie przy tworzeniu. Odtwarzacz Flash Player nie wie nic na temat warstw. Po wyeksportowaniu filmu Flasha cała zawartość umieszczona w warstwach konwertowana jest na **głębokości** (*depths*). W założeniach głębokości przypominają warstwy, ale są dostępne programistycznie. Flash zezwala na tylko jeden obiekt na danej głębokości. Jeżeli dodasz zawartość do głębokości, która już ją zawiera, istniejąca zawartość zostanie usunięta.

ActionScript pozwala na dodawanie zawartości za pomocą kodu programu. Ponieważ taka zawartość dodawana jest już w czasie działania programu, musisz skorzystać z głębokości, a nie warstw. Metody, takie jak `attachMovie()` czy `createTextField()`, które dodają zawartość w czasie wykonywania skryptu, wymagają określenia głębokości. Głębokości określane są za pomocą **liczb całkowitych**. Im wyższa ich wartość, tym wyżej będzie umieszczony obiekt. Przykładowo obiekt mający wartość głębokości *2* umieszczony będzie przed obiektem mającym wartość *1*. Zawartość dodawana przy tworzeniu filmu (na warstwach listwy czasowej) umieszczana jest na głębokościach rozpoczynających się od wartości *-13683*. Oznacza to, że zawartość ta będzie widoczna za zawartością dodaną programowo (oczywiście, przy założeniu, że zdefiniujesz jej głębokość jako wartości dodatnie).

Każdy klip filmowy posiada **własny zestaw głębokości**. Jeżeli np. korzystasz z dwóch instancji symboli klipów filmowych (*A* i *B*) i każdy z nich zawiera zagnieżdżone instancje klipów, wartości głębokości dla całych klipów *A* i *B* będą miały wpływ na to, w jaki sposób ich zawartość będzie wyświetlana względem siebie. Jeżeli *A* będzie miał wyższą wartość głębokości, cała jego zawartość będzie wyświetlana przed zawartością *B* nawet wtedy, jeśli głębokości klipów zagnieżdżonych w klipie *B* będą miały wyższą wartość głębokości niż zawartość klipu *A*. Oznacza to także, że będziesz mógł dodawać zawartość do klipu *A* na głębokości *1* oraz zawartość do klipu *B* na głębokości *1* bez usuwania pierwszej z nich.

Flash korzysta także z koncepcji **poziomów** (*levels*). Domyślnie dla całego filmu Flasha istnieje jeden poziom nazwany `_level0`. Możesz dodawać kolejne poziomy, wczytując zawartość za pomocą klasy `MovieClipLoader` lub starszej funkcji globalnej `loadMovieNum()`. Generalnie nie zaleca się korzystania z poziomów, są one pozostałością po czasach, gdy Flash nie wykorzystywał jeszcze głębokości.

Dostosowywanie środowiska programistycznego

W czasie pracy z książką będziesz korzystał ze środowiska IDE w celu wpisywania kodu zawartego w przykładach. Pisząc kod ActionScript w środowisku Flasha, na pewno docenisz możliwości dostosowania sposobu formatowania kodu i to, jak Flash pomaga przy dodawaniu kodu. W następnych podrozdziałach zapoznasz się z panelem *Actions* i możliwościami dostosowywania go.

Panel Actions

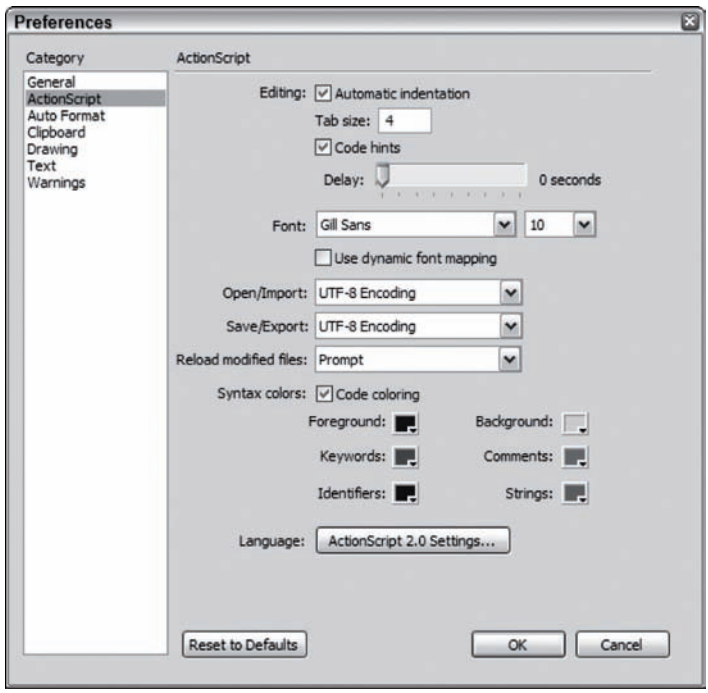
Gdy chcesz dodać kod ActionScript do dokumentu Flasha, będziesz musiał skorzystać z panelu *Actions*. Możesz go otworzyć, wybierając *Window/Actions* lub naciskając klawisz *F9*. Za pomocą panelu *Actions* możesz dodać kod skryptu do klatki kluczowej lub instancji symbolu. Skrypt zostanie dodany do klatki kluczowej, jeśli zaznaczyłeś ją na listwie czasowej, lub do instancji symbolu, jeżeli zaznaczyłeś go na scenie. Dodawanie kodu do instancji symboli nie jest już zalecane. Przed ukazaniem się Flasha w wersji MX był to jedyny sposób na dodanie niektórych elementów kodu. Dodawanie kodu do instancji symboli jest niekorzystne, gdyż wymaga o wiele więcej kodu, pogarsza skalowalność oraz może utrudnić zapanowanie nad tym, gdzie umieściłeś kod. Z powyższych przyczyn zalecane jest, aby zawsze, dodając kod do dokumentu Flasha, dodawać go do klatek kluczowych. Częstym błędem popełnianym zarówno przez początkujących, jak i ekspertów jest przypadkowe dodanie kodu do instancji symbolu zamiast (jak mieli zamiar) do klatki kluczowej. Za każdym razem, gdy dodajesz kod za pomocą panelu *Actions*, upewnij się, że na jego pasku tytułowym wyświetlone jest *Actions — Frame*.

Po lewej stronie panelu znajduje się okno narzędzi oraz nawigator skryptu. Oba narzędzia rzadko przydają się komuś, kto chce wyjść poza poziom podstawowy w nauce ActionScript. Po prawej stronie znajduje się panel skryptu, w którym można bezpośrednio wpisywać kod ActionScript. Panel skryptu stanowi w zasadzie zwykły edytor tekstowy.

Ustawienia ActionScript

Teraz dostosujesz panel *Actions* do własnych preferencji. Okno dialogowe *Preferences* możesz otworzyć, wybierając *Edit/Preferences* (rysunek 1.4).

Rysunek 1.4.



W poniższym podrozdziale zapoznasz się z niektórymi ustawieniami ActionScript, dostępnymi w kategorii *ActionScript* znajdującej się na liście w lewej części okna dialogowego *Preferences*.

Automatic indentation (automatyczne wcięcia)

W blokach kodu umieszcza się zwykle **wcięcia**, aby kod stał się bardziej czytelny. Bloki kodu są oznaczone nawiasami klamrowymi { i }. Kiedy wstawiasz lewy nawias ({) i nowy wiersz, Flash może automatycznie wstawić wcięcie w kolejnym wierszu. Opcja *Automatic indentation* (automatyczne wcięcia) jest domyślnie zaznaczona; jeżeli nie chcesz, aby do kodu były automatycznie dodawane wcięcia, usuń to zaznaczenie.

Tab size (tabulatory)

Domyślna wielkość tabulatora równa się czterem spacjom. Możesz zmienić tę wartość, jeżeli chcesz, aby przy każdym wciśnięciu klawisza *Tab* tworzone były mniejsze lub większe wcięcia.

Code hints (wskazówki kodu)

Za pomocą **wskazówek kodu** Flash może Ci pomóc, wyświetlając listę dostępnych opcji odpowiednich do już wpisanego kodu. Jeżeli np. wpiszesz nazwę zmiennej, którą Flash rozpozna jako klip filmowy, automatycznie wyświetli listę funkcji przypisanych klipom filmowym.

Wskazówki są wywoływane przez dodanie do nazwy zmiennej **przyrostka**, poprzedzonego znakiem podkreślenia (_). Wpisz w panelu *Actions* poniższy, przykładowy kod, aby zobaczyć, jak działają:

```
example_array.
```

Po tym, jak wpiszesz znak kropki (.), otworzy się menu podręczne zawierające listę dostępnych funkcji dla obiektu będącego tablicą (*array*). Pełna lista przyrostków wykorzystywanych przez Flash zawarta jest w dokumencie „About Using Suffixes To Trigger Code Hints", dostępnym w panelu *Help*.

Jakkolwiek użycie przyrostków sprawdza się jako metoda wywoływania wskazówek, to jednak nie jest to aktualnie zalecany sposób. Od wersji ActionScript 2.0 możesz do wywoływania wskazówek używać **ścisłej kontroli typów**. Ścisła kontrola typów informuje program, do jakiego typu danych odnosi się zmienna, co pozwala na wyświetlenie wskazówek kodu bez względu na to, czy do nazwy zmiennej dodany został przyrostek. Wpisz w panelu *Actions* poniższy, przykładowy kod:

```
var example:Array;
example.
```

Tak jak poprzednio, pokaże się lista metod możliwych do użycia na obiekcie typu `Array` (tablica). Jednak w tym przypadku zastosowanie przyrostka nie było konieczne. Więcej na temat składni języka ActionScript, deklarowania zmiennych i ścisłej kontroli typów dowiesz się w następnych rozdziałach.

Możesz także określić, jak długi czas upłynie do wyświetlenia wskazówki (suwak *Delay* — opóźnienie). Opóźnienie dotyczy czasu, jaki upłynie pomiędzy wywołaniem wskazówki (którym zwykle będzie wpisanie kropki po nazwie zmiennej) a jej wyświetleniem. Ustawienie małego opóźnienia może powodować, iż wskazówki będą wyświetlane nawet wtedy, kiedy nie będziesz ich potrzebował. Jeżeli natomiast będzie zbyt duże, będziesz musiał czekać na pokazanie się wskazówki. Domyślna wartość to *0* (wartości wyrażone są w sekundach), co oznacza, iż wskazówki pojawią się natychmiast po wywołaniu.

Font (czcionka)

Korzystanie z czcionki określonego rodzaju zwykle zależy od osobistych upodobań, jednak przy pisaniu kodu nie zaleca się używania niektórych czcionek. Przykładowo w czcionce *Georgia* mała litera *o* wygląda identycznie jak cyfra *0*. Może to znacznie utrudnić znajdowanie ewentualnych błędów w kodzie oraz po prostu zmniejszyć jego czytelność. Powinieneś unikać takich czcionek.

Generalnie rzecz biorąc, czcionki z rodziny **szeryfowych** (*serif fonts*) zawierają litery wyraźniej odróżniające się od cyfr oraz lepiej wyróżniają pojedyncze słowa. Także czcionki **monotypiczne** (*monospace fonts*) lepiej nadają się do wpisywania kodu. Z tych właśnie przyczyn czcionka, której Flash używa domyślnie dla kodu ActionScript, to monotypiczna czcionka z rodziny szeryfowych (konkretnie *Courier* lub *Courier New*).

Warto samemu poeksperymentować lub, jeżeli wcześniej używałeś już jakiegoś edytora i przyzwyczaiłeś się do czcionki, korzystać z niej również tutaj. Jeśli nie masz szczególnych upodobań, wypróbuj różne czcionki. Środowisko, w którym będziesz pisał skrypty, powinno być jak najbardziej przyjazne, ułatwiające pracę oraz dostosowane do Twoich preferencji.

Color (kolor)

Jednym z podstawowych sposobów, w jaki możesz dostosować panel *Actions*, jest ustawienie kolorów tła (pole *Background*), słów kluczowych (pole *Keywords*), wartości łańcuchowych (pole *Strings*), komentarzy (pole *Comments*) i innych. Dopasowując kolory do własnych preferencji, możesz zmniejszyć poziom zmęczenia oczu.

Jeżeli spędzasz wiele godzin, pisząc kod na białym tle, i nie możesz patrzeć na ekran bez mrużenia oczu, możesz zmienić kolor tła na jasnoszary (szesnastkowa wartość `#E0DBC7`).

W kodzie występuje niewiele słów kluczowych, możesz wyróżnić je kolorem niebieskim (`#0066CC`).

Słowa identyfikatory to nazwy klas oraz typów danych używanych przy deklarowaniu zmiennych. Pojawiają się w kodzie bardzo często, tak więc ustawiłem dla nich (pole *Identifiers*) kolor czerwony (`#990000`).

Kolor pierwszoplanowy (pole *Foreground*) odnosi się do tekstu stanowiącego inne elementy kodu, takie jak tworzone funkcje i metody nieprzypisane do określonych obiektów (np. `clear()`, ale już nie `this.clear()`). Domyślny kolor to kolor czarny, jednak możesz wybrać kolor szary (`#333333`), jeżeli wolisz mniejszy kontrast.

Komentarze to swoiste „znaki drogowe” kodu. Mogą ostrzec przed zawiłościami kodu, a także wyróżnić jego fragmenty składające się na większy projekt. Możesz ustawić ich kolor (pole *Comments*) na pomarańczowy (`#CC6600`).

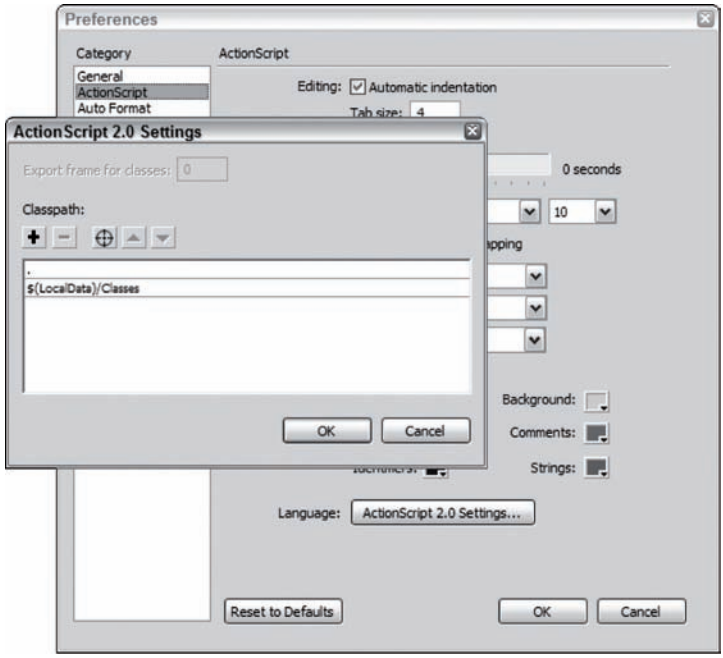
Łańcuchy znaków to przypadek szczególny. Ważne jest czasem, aby, tylko pobieżnie patrząc na kod, rozróżnić `this.n.clear()` i `this.n = "clear";`. Użyjmy koloru zielonego (`#009900`) dla łańcuchów (pole *Strings*), ponieważ łatwo go odróżnić pośród innych kolorów.

Możesz, oczywiście, wybrać własny zestaw kolorów i zmienić go w każdej chwili za pomocą okna dialogowego *Preferences*.

ActionScript 2.0 Settings (ustawienia ActionScript 2.0)

W części *Language* (język) w dolnej części okna dialogowego *Preferences* znajduje się przycisk ustawień języka ActionScript 2.0. Kliknięcie go spowoduje otwarcie się nowego okna dialogowego (rysunek 1.5), w którym możemy przypisać wartości właściwości *classpath*.

Rysunek 1.5.



Classpath (czyli ścieżka dostępu do klas) jest listą lokalizacji, w których Flash będzie mógł znaleźć pliki klas w czasie kompilowania. Możesz umieścić zestaw klas w dowolnym miejscu, pod warunkiem, że określisz wartość *classpath* dla kompilatora. Oznacza to, że będziesz mógł tworzyć biblioteki kodu do wielokrotnego wykorzystania w różnych projektach.

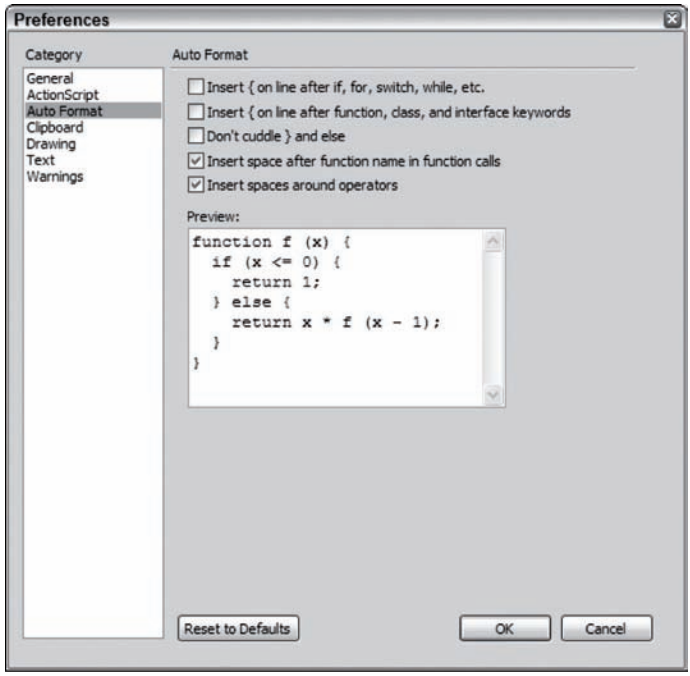
Lista wartości właściwości *classpath* powinna zawierać ścieżkę domyślną, prowadzącą do zestawu klas załączonego do programu Flash. Nie usuwaj tej domyślnej wartości. Pierwszą pozycją na liście ścieżek jest po prostu kropka (.). Oznacza to, iż Flash będzie szukał klas (do których odniesienia znajdują się w kodzie) także w folderze, w którym znajduje się dokument Flasha.

Flash szuka klas w takiej kolejności, w jakiej ułożona jest lista katalogów. Przy kompilacji wykorzysta pierwszą napotkaną instancję danej klasy. Klikając przycisk oznaczony znakiem plus (+), możesz dodawać do listy nowe katalogi. Istniejącą pozycję usuniesz z listy, zaznaczając ją, a następnie klikając przycisk ze znakiem minus (-). Za pomocą klawiszy oznaczonych strzałkami zmienisz kolejność katalogów na liście.

Auto Format (automatyczne formatowanie kodu)

Jeżeli klikniesz przycisk *Auto Format* w panelu *Actions*, do wpisywanego kodu będą automatycznie dodawane wcięcia i niektóre znaki. Możesz określić zasady tych modyfikacji. Na początku przejdź do kategorii *Auto Format* w oknie dialogowym *Preferences*, aby wyświetlić odpowiednią stronę właściwości (rysunek 1.6).

Rysunek 1.6.



Pierwsze dwa ustawienia określają, gdzie Flash będzie wstawiał nawiasy klamrowe. Jeżeli je zaznaczysz, nawias klamrowy będzie przenoszony do nowego wiersza, jeżeli natomiast pozostawisz je bez zaznaczenia, nawias pozostanie na końcu bieżącego wiersza. Jeśli nigdy wcześniej nie miałeś kontaktu z ActionScript i na razie nie wiesz, o co tu chodzi, nie martw się, gdy zapoznasz się z funkcjami i wyrażeniami, zrozumiesz wszystko.

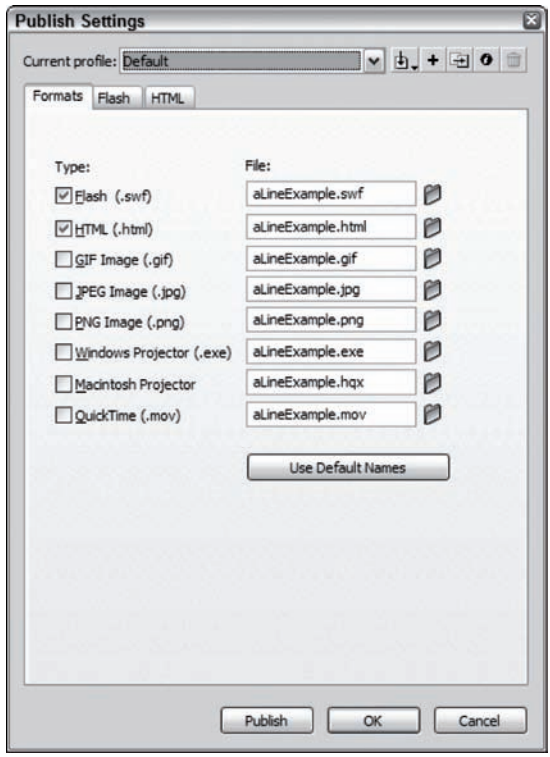
W tej chwili staraj się tylko pamiętać, gdzie zmienić poszczególne ustawienia. Najprawdopodobniej wrócisz do nich, kiedy zaczniesz tworzyć kod ActionScript i wyrobisz sobie określone preferencje pracy. Przykłady kodu, zamieszczone w niniejszej książce są — generalnie rzecz biorąc — sformatowane bez wykorzystania którejkolwiek z opcji automatycznego formatowania. Domyślne ustawienia sprawiają, iż kod jest czytelny i z łatwością rozumiany także przez innych programistów.

Publikacja projektu

Po ukończeniu projektu będziesz chciał go opublikować. Flash zawiera polecenie *Publish* (publikuj), które pozwala na wyeksportowanie projektu do wielu formatów, takich jak *SWF*, *Quicktime*, a nawet pliki wykonywalne. Okno dialogowe *Publish Settings* (ustawienia publikacji) pozwala na wybranie formatów, w których opublikujesz projekt, a także dostosowanie ustawień dla każdego formatu. Okno dialogowe daje również możliwość automatycznego skonstruowania skryptu JavaScript wykrywającego odtwarzacz Flash Player oraz obrazów zastępczych lub dokumentów HTML do wykorzystania w sytuacji, gdy odtwarzacz nie zostanie wykryty. Panel *Publish Settings* pomoże Ci także radzić sobie ze specjalnymi odtwarzaczami, takimi jak Flash Lite i odtwarzacz dla urządzeń typu Pocket PC.

Okno dialogowe *Publish Settings* (rysunek 1.7) pozwala na zapisywanie i zachowywanie własnych profilów ustawień publikacji. Może się to okazać przydatne, gdy chcesz, aby kilku programistów używało jednolitych ustawień lub masz zamiar korzystać z jednakowego profilu we wszystkich swoich projektach.

Rysunek 1.7.



W oknie dialogowym *Publish Setting* możesz określić formaty, w jakich chcesz opublikować projekt. Domyślnie wybrane formaty to Flash oraz HTML. Kiedy wybierasz polecenie *Publish* (*File/Publish*), Flash wyeksportuje projekt do tych formatów, które są zaznaczone w oknie dialogowym *Publish Settings*. Możesz określić domyślne ścieżki do lokalizacji, w których będą zapisywane publikowane projekty, w tym również odrębną ścieżkę dla każdego formatu. Ustawienia publikacji są zapisywane wraz z zapisem dokumentu Flasha.

Dla każdego nowego formatu, który wybierzesz, pojawia się nowa zakładka (z wyjątkiem formatów *Windows Projector* i *Macintosh Projector*, które nie mają dodatkowych ustawień). Domyślnie okno dialogowe *Publish Settings* ma trzy zakładki — *Formats*, *Flash* i *HTML*.

Zakładka Flash

W zakładce *Flash* w oknie dialogowym *Publish Settings* określone są właściwości dla plików w formacie SWF. Opcja *Version* (wersja) podaje, które wersje odtwarzacza będą mogły wyświetlać aplikację. Powinieneś wybrać tę, która będzie najbardziej odpowiednia dla potencjalnych odbiorców i Twojej aplikacji. Jeżeli np. Twoja aplikacja korzysta z opcji wygładzania (renderowania) tekstu, dostępnej we Flashu 8, wymagane będzie użycie odtwarzacza

w wersji Flash 8. Jeśli jednak tworzysz tylko prostą animację, być może wystarczy Flash 5. Kiedy dokonujesz publikacji dokumentu w formacie FLA do formatu SWF i kod w publikowanym dokumencie nie będzie działał prawidłowo w odtwarzaczu w wersji określonej w panelu *Publish Settings*, wyświetli się ostrzeżenie

Opcja *Load order* (kolejność ładowania) określa, w jaki sposób będzie renderowana pierwsza klatka aplikacji. Ten proces będzie widoczny głównie na wolniejszych komputerach i możesz rozważyć, jaki nadać mu wygląd. Domyślna opcja *Bottom up* (z dołu do góry) jest, w większości przypadków, wystarczająca.

Ustawienie w polu *ActionScript version* (wersja ActionScript) powinno odpowiadać wersji języka, którą planujesz wykorzystać w Twojej aplikacji. Chociaż w wersji ActionScript 2.0 możesz korzystać z konstrukcji kodu z innych wersji, powinieneś zmienić to ustawienie, aby ostrzec innych programistów, którzy mogą otworzyć w przyszłości Twój dokument, jakiego rodzaju składni użyłeś. To ustawienie ma wpływ na działanie kompilatora, dyrektywy kompilatora, sprawdzanie poprawności oraz ocenę zgodności ze ścisłą kontrolą typów.

Kolejna grupa opcji pozwala na ustawienie atrybutów wynikowego pliku SWF. Jeżeli zaznaczysz pole *Protect from import* (chroń przed zaimportowaniem) jako atrybut dla pliku SWF, zostanie uaktywniona opcja pozwalająca na ochronę pliku SWF za pomocą hasła (pole *Password*). Takie ustawienie będzie chronić Twój plik SWF przed próbami załadowania go do pliku FLA lub większej aplikacji.

Opcja *Debugging permitted* (zezwalaj na wykrywanie błędów) zezwala na dostęp do pliku SWF programowi wykrywającemu błędy. Nie zaznaczaj tej opcji przy tworzeniu ostatecznego pliku SWF, jej istnienie jest swoistym *faux pas* ze strony twórców programu, gdyż nie ma ona wpływu na kształt aplikacji dla zwykłych użytkowników.

Opcja *Compress movie* (kompresuj film) to przydatne ustawienie, pozwalające na zmniejszenie wynikowego pliku SWF o 10, 20 lub nawet 30%. Odtwarzacz SWF ma wbudowany dekompresor uaktywniający się w momencie załadowania pliku SWF. Atrybut ten jest szczególnie przydatny dla złożonych aplikacji, zwierających dużo grafiki, kodu ActionScript oraz innej wbudowanej zawartości, w mniejszym stopniu dla w pełni dynamicznych plików SWF, w których niemal cała zawartość jest ładowana w czasie wykonywania.

JPEG quality (jakość kompresji JPEG) to często pomijane ustawienie, które może również przyczynić się do zmniejszenia rozmiaru pliku SWF. Będzie miało wpływ na wszystkie bitmapy, które nie mają przypisanych indywidualnych ustawień jakości. Indywidualne ustawienia jakości dostosujesz za pomocą panelu *Properties* dla każdej bitmapy znajdującej się w bibliotece.

Ostatnie dwa ustawienia dotyczą tylko dźwięków umieszczonych na listwie czasowej bezpośrednio z biblioteki.

Zakładka HTML

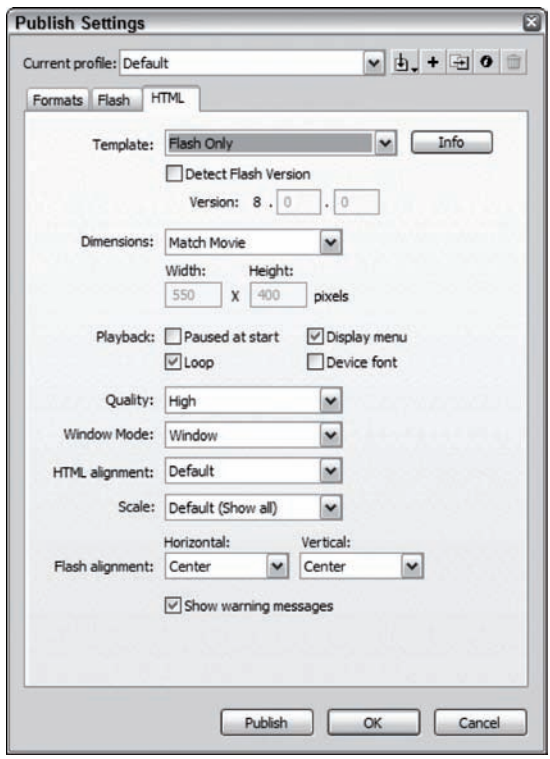
Publikując plik SWF, warto zdefiniować jego otoczenie w formacie strony HTML. Chociaż większość przeglądarek będzie bezpośrednio odtwarzać plik SWF za pomocą odpowiedniego modułu dodatkowego, to jednak trudno przewidzieć, czy wersja Flasha zainstalowana na

komputerze użytkownika jest wystarczająca do odtworzenia zawartości w formacie SWF. Dokument HTML może zawierać skrypt JavaScript i odpowiednie znaczniki `<object>` służące do automatycznego sprawdzania poprawności wersji odtwarzacza Flasha.

Możesz także wybrać opcje HTML dla określonego urządzenia lub przeglądarki. Przykładowo urządzenia przenośne używają własnego zestawu znaczników do dołączania obiektów, takich jak moduł dodatkowy, do strony. Podobnie jak w przypadku ustawień wersji odtwarzacza Flasha powinieneś dokładnie rozważyć zgodność dokumentów HTML z systemami użytkowników końcowych. Kolejna część — *Detect Flash Version* (wykrywaj wersję Flasha) — służy do włączenia wykrywania wersji Flasha za pomocą prostego skryptu JavaScript.

W panelu *HTML* okna dialogowego *Publish Settings* (rysunek 1.8) możesz także ustawić opcje odtwarzania (część *Playback*) oraz atrybuty znaczników `<embed>` i `<object>`. Oczywiście, możesz także utworzyć własny dokument HTML i dowolnie zmienić atrybuty.

Rysunek 1.8.



Tak, jak w przypadku wszystkich innych ustawień w oknie dialogowym *Publish Settings*, powinieneś przetestować każdy wybór i przekonać się, jaki ma wpływ na działanie aplikacji i kompatybilność z różnymi przeglądarkami.

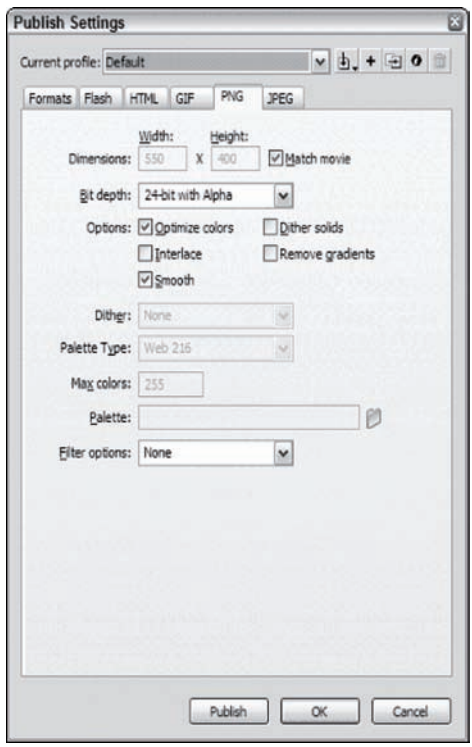
Jest także możliwe tworzenie własnych szablonów dokumentów HTML.

Zakładki dotyczące obrazów

Jeżeli wrócisz do zakładki *Formats* w oknie dialogowym *Publish Settings*, zobaczysz, że przy kompilowaniu pliku SWF możliwe jest także tworzenie obrazów w trzech formatach. Każdy z trzech typów plików ma własne opcje służące do ustawiania atrybutów tworzonych obrazów, w tym wymiarów, jakości, wygładzania i innych. W tym przypadku także najlepiej sprawdzić metodą prób i błędów, jaki wpływ na wygląd tworzonego obrazu mają poszczególne ustawienia. Przy publikacji obrazów celem jest osiągnięcie jak najlepszej jakości przy jak najmniejszej objętości pliku i ustalenie kompromisu zależnego od zawartości obrazu.

Na rysunku 1.9 pokazujemy przykładowy wygląd zakładki PNG w panelu *Publish Settings*.

Rysunek 1.9.



Opcje wykrywania odtwarzacza Flash Player

Utworzyłeś już plik SWF, zatem będziesz chciał, aby wynik Twojej pracy mógł być oglądany w tak wielu konfiguracjach systemowych, jak to tylko możliwe. Odtwarzacz Flash Player możesz wykrywać na kilka sposobów. Najlepszym, a jednocześnie najprostszym z nich jest wykorzystanie zestawu skryptów wykrywających Flasha (*Flash Detection Kit*). We Flashu 8 zestaw ten otwieramy za pomocą opcji w oknie dialogowym *Publish Settings* w zakładce HTML. Możliwe jest także oddzielne pobranie zestawu, co będzie pomocne podczas włączania go do stron wykorzystujących serwlety Javy oraz starszych stron HTML.

Domyślny szablon publikacji w formacie HTML — *Flash Only* — dostępny w oknie dialogowym *Publish Settings*, będzie wyświetlał zawartość projektu tylko wtedy, gdy wiadomo, że w systemie jest już zainstalowany odtwarzacz Flash Player. Szablon ten jest odpowiedni do lokalnego testowania plików SWF. Jednak niedołączenie własnego mechanizmu wykrywania spowoduje pozostawienie tego zadania przeglądarce. Internet Explorer w takim przypadku wyświetla ostrzeżenie z pytaniem, czy chciałbyś zainstalować lub uaktualnić odtwarzacz Flash Player, a następnie rozpoczyna prosty proces instalacji oparty na certyfikacie. Wielu użytkowników, w obawie przed szkodliwym oprogramowaniem (lub z innych przyczyn), odrzuci prośbę o akceptację certyfikatu. W innych przeglądarkach cały proces informowania o instalacji nakładki ogranicza się do wyświetlenia prostej wiadomości bez możliwości jej odrzucenia lub nawet bez informacji, gdzie można dowiedzieć się więcej na temat nakładek. Trudno zaakceptować taki sposób, a tym bardziej uznać go za przyjazny dla użytkownika.

Użycie zestawu wykrywania Flasha sprawi, iż strona HTML zostanie wyeksportowana wraz z odpowiednimi skryptami JavaScript otaczającymi plik SWF. Jak skrypt stwierdzi, że w systemie zainstalowany jest odtwarzacz i jego wersja jest poprawna, za pomocą metody języka JavaScript `document.write` dopisze do kodu strony odpowiednie znaczniki `<object>` i `<embed>`.

Gdy wersja odtwarzacza Flash Player okaże się zbyt stara, by odtworzyć Twój pliku SWF, zostanie wykorzystana alternatywna metoda, która dopisze do kodu znacznik `<image>` i wskaże odpowiedni obraz w formacie *.gif*.

Najczęściej pozwalamy Flashowi na dołączenie zestawu skryptów, a następnie edytujmy otrzymany plik HTML tak, aby działał w pożądany przez nas sposób. W niektórych przypadkach zastąpienie pliku SWF obrazem GIF jest dopuszczalne, ale — oczywiście — nie w przypadku całych aplikacji w formacie SWF. Wtedy lepiej skorzystać z metody języka JavaScript `document.write` do dołączenia w kodzie HTML instrukcji, jak poradzić sobie z uaktualnieniem nakładki. Pozwoli to użytkownikom na zaktualizowanie systemu i zrozumienie, dlaczego aplikacja SWF nie działała przy pierwszej wizycie na stronie.

W każdym przypadku lepiej, gdy konieczne jest wykrycie rozszerzeń, od razu wyświetlić użytkownikowi to, co jest możliwe. Następnie określić odpowiednie zachowanie się strony w zależności od koniecznych uaktualnień i wreszcie przeprowadzić proces uaktualniania możliwie elegancko, bez wyświetlania zbędnych ostrzeżeń lub pustych elementów.

Podsumowanie

W tym rozdziale pokrótce zapoznałeś się ze środowiskiem IDE. Wiele innych jego właściwości opanujesz samodzielnie już w trakcie pracy. Możesz się zastanawiać, dlaczego nie opisaliśmy tu wszystkich. Niniejsza książka koncentruje się na języku ActionScript i celem tego rozdziału było wprowadzenie do środowiska IDE tak, abyś mógł wykonać ćwiczenia zawarte w całej książce. W każdym z przykładów będziesz zapoznawał się ze skrótami i metodami pracy, które nie zostały tu jeszcze omówione. Korzystanie ze skrótów klawiszowych może znacznie przyśpieszyć pracę, jednak każdy system operacyjny ma nieco inny ich zestaw. Jeżeli opisany tu skrót nie działa, spróbuj ręcznie odnaleźć w menu odpowiednie polecenie. W większości przypadków skrót klawiszowy pokazany jest po prawej stronie polecenia w menu. Dodatek B zawiera pełną listę skrótów klawiszowych.

Środowisko Flasha jest niezwykle elastyczne. Może się zdarzyć, że prezentowane sposoby pracy będą zupełnie inne niż przyjęte przez Ciebie. Właśnie z powodu, iż środowisko IDE jest tak bardzo elastyczne, chcielibyśmy zachęcić Cię do przyjęcia własnych metod pracy i samodzielnego zapoznawania się z możliwościami IDE.

Ćwiczenia

1. Utwórz nowy plik FLA (czyli dokument Macromedia Flash), wybierając *File/New*, a następnie *Flash document* z okna *New Document*. Utwórz nową warstwę, umieść na scenie rysunek wektorowy lub obraz. Dodaj do klatki prosty skrypt, wyświetlający prosty tekst, np. „Witaj Świecie!”. Zmień nazwę warstwy na `test` i zapisz plik FLA. Utwórz nowy plik FLA. Wróć do pierwszego dokumentu, kliknij prawym przyciskiem myszy klatkę zawierającą skrypt (*option*+kliknięcie w Mac OS) i wybierz *Copy*. Wklej klatkę do nowego pliku FLA. Porównaj wyniki.
2. Otwórz nowy plik FLA i otwórz jego bibliotekę. Dodaj nowy symbol (przycisk lub klip filmowy) za pomocą menu kontekstowego. Kliknij dwukrotnie właśnie utworzoną instancję symbolu i dodaj do niego trochę grafiki. Otwórz nowy plik FLA i jego bibliotekę. Wróć do biblioteki pierwotnego pliku i przeciągnij instancję obiektu do nowego pliku FLA. Zauważ, że symbol jest teraz w dwóch bibliotekach.
3. Otwórz jednocześnie kilka plików FLA. Zobacz, jak można się przemieszczać pomiędzy nimi za pomocą systemu zakładek na górze okna IDE. Zminimalizuj i zmaksymalizuj jeden z plików. Zaobserwuj, jak działa IDE w trybie zminimalizowanych okien.