



Technologia i rozwiązania

Tworzenie aplikacji dla iOS we Flashu

Receptury

100 praktycznych receptur na rozwijanie aplikacji iOS w programie
Flash Professional z użyciem Adobe AIR



Christopher Caleb



Tytuł oryginału: Flash iOS Apps Cookbook

Tłumaczenie: Joanna Zatorska

ISBN: 978-83-246-4993-8

Copyright © Packt Publishing 2012.

First published in the English language under the title 'Flash iOS Apps Cookbook'

© Helion 2013. All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Wydawnictwo HELION dołożyło wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie bierze jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Wydawnictwo HELION nie ponosi również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION

ul. Kościuszki 1c, 44-100 GLIWICE

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:

<ftp://ftp.helion.pl/przyklady/twapre.zip>

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/twapre>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

O autorze	9
Podziękowania	11
O recenzentach	13
Wstęp	15
Rozdział 1. Wstęp do tworzenia aplikacji iOS	21
Wstęp	22
Rejestracja w programie iOS Developer Program	22
Dostęp do serwisu iOS Provisioning Portal	29
Generowanie wniosku o podpisanie certyfikatu w systemie Windows	32
Generowanie wniosku o podpisanie certyfikatu w systemie MAC OS X	34
Uzyskiwanie certyfikatu programisty	35
Tworzenie certyfikatu P12 w systemie Windows	38
Tworzenie certyfikatu P12 w systemie MAC OS X	41
Rejestracja urządzenia	43
Tworzenie identyfikatora aplikacji	47
Tworzenie programistycznego profilu informacyjnego	50
Instalacja profilu informacyjnego na urządzeniu	53

Rozdział 2. Tworzenie aplikacji iOS w programie Flash	57
Wstęp	57
Instalacja AIR SDK	58
Tworzenie dokumentu AIR for iOS	60
Umieszczanie elementów na stole montażowym	63
Opis działania	66
Ogólne ustawienia środowiska AIR for iOS	66
Opis działania	68
Ustawienia wdrażania aplikacji w środowisku AIR for iOS	70
Kompilacja w programie Flash Professional	73
Instalacja aplikacji przy użyciu iTunes	75
Rozdział 3. Tworzenie pierwszej aplikacji	79
Wstęp	79
Tworzenie podstawowej klasy dokumentu	80
Zapobieganie przejściu ekranu w stan beczynności	82
Wielozadaniowość	85
Eleganckie zamykanie aplikacji	87
Dołączanie klas do symboli klipów filmowych	89
Korzystanie z pętli uaktualnień	92
Dołączanie obrazu rozruchowego aplikacji	96
Dołączanie ikon	99
Edycja pliku deskryptora aplikacji	101
Zdalne debugowanie	104
Rozdział 4. Konwersja projektów Flash na platformę iOS	109
Wstęp	109
Interakcja z użytkownikiem	110
Zapisywanie stanu aplikacji	115
Splaszczanie listy wyświetlania	118
Konwersja grafiki wektorowej na rastrową	121
Zmiana rozmiarów bitmap	126
Maskowanie obiektów	129
Praca z zewnętrznymi plikami SWF	132
Rozdział 5. Technologia Multi-touch i gesty	137
Wstęp	137
Ustawianie trybu interakcji z punktem kontaktu	138
Wykrywanie kilku punktów kontaktu	141
Przeciąganie kilku wyświetlonych obiektów	146
Śledzenie ruchu	150
Ustawianie trybu interakcji z użyciem gestów	152
Obsługa gestu machnięcia	156
Przewijanie obiektu	159
Obracanie obiektu	162
Powiększanie i zmniejszanie obiektu	164

Rozdział 6. Akceleracja sprzętowa grafiki	167
Wstęp	167
Porównanie grafiki wektorowej i rastrowej	168
Tryb GPU-Blend	171
Tryb GPU-Vector	176
Buforowanie bitmap	181
Buforowanie z użyciem macierzy	187
Dostęp do bitmap w kodzie ActionScript	192
Wczytywanie bitmap w trakcie działania programu	196
Praca z zestawami sprite'ów	200
Animowanie bitmap z użyciem ActionScript	205
Rozdział 7. Praca z tekstem i wirtualną klawiaturą	209
Wstęp	209
Używanie czcionek urządzenia w polach tekstowych	210
Korzystanie z osadzonych czcionek w polach tekstowych	213
Wejściowe pole tekstowe	217
Pobieranie tekstu	221
Konfigurowalne przewijanie podczas aktywacji wirtualnej klawiatury	223
Uaktualnianie dynamicznych pól tekstowych	227
Użycie natywnych kontroltek tekstowych iOS	231
Rozdział 8. Zmiany rozdzielczości i orientacji ekranu	237
Wstęp	237
Wybór docelowego urządzenia	238
Dostosowywanie do ekranów Retina	241
Obsługa urządzeń o różnej rozdzielczości ekranu	246
Ustawianie domyślnego formatu ekranu	251
Włączanie automatycznej orientacji	252
Nasłuchiwanie na zmiany orientacji	255
Reakcja na zmiany orientacji	258
Rozdział 9. API geolokalizacji i akcelerometru	263
Wstęp	263
Określanie bieżącego położenia	264
Określanie prędkości i kierunku poruszania się	269
Sprawdzanie dostępności geolokalizacji	273
Reagowanie na zmiany zarejestrowane przez akcelerometr	277
Wykrywanie potrząsania	282
Rozdział 10. Obsługa kamery i mikrofonu	287
Wstęp	287
Zapisywanie w rolce z aparatu	288
Odczyt rolki z aparatu	291
Fotografowanie z użyciem domyślnej aplikacji	296

Praca z wbudowanymi aparatami	300
Nagrywanie dźwięku z użyciem mikrofonu	304
Odtwarzanie nagranych dźwięku	309
Rozdział 11. Wyświetlanie stron WWW	315
Wstęp	315
Wyświetlanie strony WWW w Safari	316
Renderowanie strony WWW w aplikacji	318
Przeglądanie historii wyświetlanych stron	321
Renderowanie lokalnej strony WWW	325
Dynamiczne generowanie lokalnej strony WWW	328
Tworzenie zrzutu strony WWW	334
Rozdział 12. Praca z wideo i dźwiękiem	337
Wstęp	337
Odtwarzanie lokalnego wideo w formacie FLV	338
Odtwarzanie lokalnego wideo w formacie H.264	343
Kontrola odtwarzania wideo	348
Osadzanie dźwięku	353
Odtwarzanie osadzonego dźwięku	355
Strumień dźwięku	358
Kontrola odtwarzania dźwięku	362
Rozdział 13. Łączność, ciągłość i schematy URI	367
Wstęp	367
Monitorowanie połączenia z internetem	368
Ustawianie stałego połączenia Wi-Fi	372
Dostęp do własnych katalogów aplikacji	374
Zapisywanie plików	377
Odczytywanie plików	381
Uruchamianie aplikacji systemowych	385
Uruchamianie App Store	389
Uruchamianie aplikacji Mapy	391
Określanie wymagań sprzętowych	395
Rozdział 14. Używanie natywnych rozszerzeń i ADT	399
Wstęp	399
Kompilacja w wierszu poleceń w systemie Windows	400
Kompilacja w wierszu poleceń w systemie Mac OS X	406
Korzystanie z natywnego rozszerzenia	411
Tworzenie pakietu z natywnym rozszerzeniem	416
Włączanie trybu Interpreter Mode	420
Uruchamianie ADT poza katalogiem instalacyjnym programu Flash Professional CS5	424

Rozdział 15. Optymalizacja kodu ActionScript	429
Wstęp	429
Deklarowanie typów danych	430
Zastępowanie typu Object własnymi klasami	432
Optymalizacja pętli	434
Zastępowanie tablic wektorami	437
Ponowne wykorzystanie instancji	440
Korzystanie ze statycznych zmiennych i metod	443
Korzystanie z metod wywołania zwrotnego	446
Wykorzystanie propagacji zdarzeń	449
Dodatek A. Receptury przeznaczone dla programu Flash Professional CS5.5	455
Dodatek B. Wyniki pomiarów optymalizacji kodu ActionScript	457
Deklarowanie typów danych	458
Zastępowanie typu Object własnymi klasami	460
Optymalizacja pętli	462
Zastępowanie tablic wektorami	464
Skorowidz	469

Zespól

Author

Christopher Caleb

Reviewers

Daniel Albu
JP Berrie
Simon Buckley
Mark Doherty
Richard England
Stuart McLeman
Brian Rinaldi
David Wagner

Acquisition Editor

Douglas Paterson

Lead Technical Editor

Dayan Hyames

Technical Editor

Kedar Bhat

Copy Editor

Neha Shetty

Project Coordinator

Alka Nayak

Proofreader

Mario Cecere

Indexer

Rekha Nair

Graphics

Valentina D'Silva
Manu Joseph

Production Coordinator

Arvindkumar Gupta

Cover Work

Arvindkumar Gupta

API geolokalizacji i akcelerometru

W tym rozdziale omówione zostaną następujące zagadnienia:

- Określanie bieżącego położenia.
- Określanie prędkości oraz kierunku.
- Sprawdzanie dostępności geolokalizacji.
- Reagowanie na zmiany wykryte przez akcelerometr.
- Wykrywanie potrząsania.

Wstęp

Rodzina urządzeń iOS używa wielu wbudowanych czujników, włącznie z trójosiowym akcelerometrem, kompasem cyfrowym, aparatem, mikrofonem oraz odbiornikiem GPS. Ich dołączenie do urządzeń otwarło przed programistami ogromne możliwości, co doprowadziło do powstania wielu innowacyjnych i ciekawych aplikacji, które przyczyniły się do ogromnego sukcesu App Store.

W tym rozdziale skupimy się na dwóch z najbardziej popularnych czujników — na akcelerometrze oraz odbiorniku GPS; obydwa są obsługiwane przez środowisko AIR for iOS.

Akcelerometr rejestruje przyspieszenie i pozwala na wykrycie fizycznej orientacji, ruchu i wibracji. Mimo że na początku mógł się on wydawać gadżetem, wraz z rozwojem platformy pojawiały się nowe, coraz bardziej wyrafinowane sposoby jego wykorzystania. Od gier, poprzez zdalne sterowanie, pakiety graficzne, aż po generowanie dźwięku — akcelerometr stał się środkiem komunikacji z użytkownikiem w wielu aplikacjach.

Urządzenia takie jak iPhone rozpoznają również swoją lokalizację. Wbudowany w nie odbiornik GPS pozwala określić położenie w dowolnym miejscu na świecie. Ponadto możliwe stało się śledzenie ruchu, określanie bieżącej prędkości, a nawet kierunku, w jakim zwrócone jest urządzenie. Oprócz określania położenia geograficznego serwisy lokalizacyjne znajdują zastosowanie w różnego rodzaju obszarach, począwszy od fotografii, a skończywszy na aplikacjach służących do komunikacji.

AIR udostępnia API, które umożliwia skorzystanie z danych zarejestrowanych przez akcelerometr oraz odbiornik GPS. Obsługę akcelerometru zapewnia klasa `flash.sensors.Accelerometer`, natomiast klasa `flash.sensors.Geolocation` pozwala na tworzenie aplikacji wykorzystujących informacje o położeniu.

Określanie bieżącego położenia

Urządzenia z rodziny iOS rozpoznają swoje położenie, co pozwala wyznaczyć przybliżone współrzędne geograficzne użytkownika. Sposób, w jaki można to osiągnąć, zależy od sprzętu zainstalowanego na urządzeniu. Na przykład pierwszy model iPhone'a, wszystkie modele iPoda touch oraz iPady uzyskują informację o położeniu na podstawie triangulacji sieci Wi-Fi. Pozostałe urządzenia mogą dokładniej obliczyć położenie, korzystając z wbudowanego odbiornika GPS lub triangulacji wież przekaźnikowych telefonii komórkowej.

Środowisko AIR SDK udostępnia warstwę abstrakcji, która pozwala na uzyskanie informacji na temat położenia w sposób niezależny od dostępnego sprzętu. Oznacza to, że ten sam kod pozwoli na dostęp do danych geograficznych zarejestrowanych przez dowolne urządzenie iOS.

Dzięki tej recepturze nauczysz się, jak określić bieżące położenie.

Przygotowanie

Na potrzeby tej receptury przygotowano wstępny projekt. W programie Flash Professional wyświetl plik *recipe fla* znajdujący się w katalogu *rozdzial09\receptura1* w pakiecie dołączonym do książki.

Na stole montażowym znajduje się sześć dynamicznych pól tekstowych. Zapełnimy każde z nich danymi o położeniu zarejestrowanymi przez urządzenie.

Jak to zrobić...

Wykonaj następujące czynności, aby pobrać i wyświetlić dane geolokalizacyjne:

1. Utwórz klasę dokumentu i nazwij ją `Main`.
2. Zaimportuj następujące klasy i utwórz zmienną prywatną typu `Geolocation`:

```

package {
    import flash.display.MovieClip;
    import flash.events.GeolocationEvent;
    import flash.sensors.Geolocation;
    public class Main extends MovieClip {
        private var geo:Geolocation;
        public function Main() {
            // constructor code
        }
    }
}

```

3. W konstruktorze klasy zainicjalizuj obiekt `Geolocation` i rozpocznij nasłuchiwanie na jego aktualizacje:

```

public function Main() {
    if(Geolocation.isSupported)
    {
        geo = new Geolocation();
        geo.setRequestedUpdateInterval(1000);
        geo.addEventListener(GeolocationEvent.UPDATE, geoUpdated);
    }
}

```

4. Następnie napisz funkcję obsługi zdarzenia, która będzie odczytywać uaktualnione dane geolokalizacyjne i wyświetlać je w dynamicznych polach tekstowych:

```

private function geoUpdated(e:GeolocationEvent):void {
    latitudeField.text = e.latitude.toString();
    longitudeField.text = e.longitude.toString();
    altitudeField.text = e.altitude.toString();
    hAccuracyField.text = e.horizontalAccuracy.toString();
    vAccuracyField.text = e.verticalAccuracy.toString();
    timestampField.text = e.timestamp.toString();
}

```

5. Zapisz klasę w pliku *Main.as* w tym samym katalogu, w którym znajduje się projekt FLA. Zapisz również projekt.
6. Opublikuj i przetestuj aplikację na urządzeniu.
7. Po pierwszym uruchomieniu aplikacji pojawi się natywne okno dialogowe iOS z komunikatem następującej treści:

„c9 r1” chce użyć informacji o Twoim położeniu.

Stuknij przycisk **OK**, aby udostępnić aplikacji dane o położeniu urządzenia.

Urządzenia, na których zainstalowano iOS w wersji 4 lub wyższej, zapamiętają Twój wybór, natomiast urządzenia ze starszą wersją iOS będą wyświetlać ten komunikat po każdym uruchomieniu aplikacji.

Na ekranie zostaną wyświetlone dane o położeniu, które będą okresowo uaktualniane. Pospaceruj trochę z urządzeniem i sprawdź, jak będą się zmieniać wyświetlane dane podczas zmian położenia geograficznego.

Opis działania

W pakiecie `flash.sensors` środowiska AIR dostępna jest klasa `Geolocation`, która pozwala odczytać dane o położeniu zarejestrowane przez urządzenie. W tym celu należy utworzyć instancję klasy `Geolocation` i przechwycić wywoływane przez nią zdarzenie `GeolocationEvent.UPDATE`.

Dokonałiśmy tego w konstruktorze klasy dokumentu, korzystając ze zmiennej prywatnej `geo`, która przechowuje referencję do obiektu:

```
geo = new Geolocation();
geo.setRequestedUpdateInterval(1000);
geo.addEventListener(GeolocationEvent.UPDATE, geoUpdated);
```

Częstotliwość pobierania danych lokalizacyjnych można zmienić, wywołując metodę `Geolocation.setRequestedUpdateInterval()`. Widać to w powyższym fragmencie kodu, w którym zdefiniowaliśmy przedział czasu między kolejnymi odczytami równy 1000 milisekund. Jest to jedynie wskazówka dla urządzenia, ponieważ rzeczywisty czas między kolejnymi odczytami może być nieco krótszy lub dłuższy. Pominięcie wywołania tej metody spowoduje użycie domyślnego przedziału czasu, który może mieć przeróżną wartość: od kilku milisekund aż do kilku sekund, w zależności od możliwości sprzętowych urządzenia.

Każde zdarzenie `UPDATE` spowoduje utworzenie obiektu `GeolocationEvent`, którego właściwości opisują bieżące położenie. Nasza metoda `geoUpdated()` przechwytuje ten obiekt i wyświetla wartości kilku z tych właściwości w dynamicznych polach tekstowych znajdujących się na stole montażowym.

```
private function geoUpdated(e:GeolocationEvent):void {
    latitudeField.text = e.latitude.toString();
    longitudeField.text = e.longitude.toString();
    altitudeField.text = e.altitude.toString();
    hAccuracyField.text = e.horizontalAccuracy.toString();
    vAccuracyField.text = e.verticalAccuracy.toString();
    timestampField.text = e.timestamp.toString();
}
```

Wyświetlone zostaną następujące dane:

- szerokość i długość geograficzna,
- wysokość,
- dokładność pozioma i pionowa,
- znacznik czasu.

Szerokość i długość geograficzna służą do wyznaczenia Twojego położenia geograficznego. Można również uzyskać informacje o wysokości mierzonej w metrach. Podczas przemieszczania się z urządzeniem wartości te będą odświeżane zgodnie z nowym położeniem.

Wyświetlana jest również dokładność danych lokalizacyjnych, która zależy od możliwości sprzętowych urządzenia. Zarówno dokładność pionowa, jak i pozioma są mierzone w metrach.

Ostatnia wartość, czyli znacznik czasu, jest związana z każdym przechwyconym obiektem `GeolocationEvent` i pozwala na określenie rzeczywistego czasu, jaki upływa między wyzwoleniem kolejnych zdarzeń. Znacznik czasu określa liczbę milisekund, które upłynęły od czasu ostatniego uruchomienia aplikacji.

Niektóre urządzenia niemające odbiornika GPS będą dość rzadko wyzwalać zdarzenie `UPDATE`. Na początku działania aplikacji wyzwalane jest jedno lub dwa zdarzenia `UPDATE`, zaś dodatkowe będą wyzwalane jedynie po znacznej zmianie położenia.

Zwróć też uwagę na statyczną właściwość `Geolocation.isSupported` wykorzystywaną w konstruktorze. Co prawda obecnie właściwość ta jest równa `true` na wszystkich urządzeniach iOS, jednak nie możemy być tego pewni w przypadku przyszłych urządzeń. Sprawdzanie obsługi geolokalizacji jest zalecane również podczas tworzenia aplikacji na różne platformy.

Więcej informacji na ten temat znajdziesz pod hasłami `flash.sensors.Geolocation` i `flash.events.GeolocationEvent` w systemie pomocy Adobe Community Help.

Dodatkowe informacje

Ilość dostępnych informacji oraz ich dokładność zależy od możliwości urządzenia.

Dokładność

Dokładność danych lokalizacyjnych zależy od metod, jakimi posługuje się urządzenie podczas wyznaczania położenia. Zazwyczaj urządzenia iOS z odbiornikiem GPS pozwalają uzyskać lepszy rezultat niż urządzenia korzystające z triangulacji Wi-Fi.

Przykładowo po uruchomieniu aplikacji z tego ćwiczenia na urządzeniu iPhone 4, mającym odbiornik GPS, uzyskamy dokładność poziomą równą 10 metrów. Ta sama aplikacja uruchomiona na iPodzie touch trzeciej generacji, który opiera się na sieci Wi-Fi, będzie raportować poziomą dokładność równą 100 metrów. Spora różnica!

Wysokość

Bieżącą wysokość można uzyskać jedynie na urządzeniach wyposażonych w odbiornik GPS. Na pozostałych urządzeniach właściwość `GeolocationEvent.verticalAccuracy` będzie równa `-1`, zaś właściwość `GeolocationEvent.altitude` będzie równa `0`. Dokładność pionowa równa `-1` oznacza, że nie można zarejestrować wysokości.

Powinieneś być świadom tych ograniczeń i zadbać o odpowiednie przygotowanie kodu aplikacji korzystającej z danych lokalizacyjnych. Nigdy nie zakładaj, że każde urządzenie będzie spełniać wszystkie wymagania.

Jeśli Twoja aplikacja wymaga obecności odbiornika GPS, możesz to uwzględnić w pliku de-skryptora aplikacji. Dzięki temu użytkownicy, których urządzenia nie spełniają tego wymagania, nie będą pobierać Twojej aplikacji z App Store.

Więcej informacji na ten temat znajdziesz w recepturze „Określanie wymagań sprzętowych” w rozdziale 13.

Wyświetlanie położenia na mapie

Najbardziej oczywistym zastosowaniem uzyskanych danych geolokalizacyjnych jest wyświetlanie położenia na mapie. Zazwyczaj aplikacje pozyskują dane o położeniu geograficznym bieżącej lokalizacji i wyświetlają mapę otoczenia. Można to osiągnąć na kilka sposobów, jednak najłatwiejszym rozwiązaniem wydaje się uruchomienie natywnej aplikacji Mapy i przekazywanie do niej danych lokalizacyjnych. Opis tej metody znajdziesz w recepturze „Uruchamianie aplikacji Mapy” w rozdziale 13.

Jeśli wolisz rozwiązanie z użyciem ActionScript, możesz skorzystać z API UMap ActionScript 3.0, które pozwala na integrację danych kartograficznych udostępnianych przez wielu dostawców, takich jak Bing, Google czy Yahoo! API można pobrać po zarejestrowaniu się na stronie www.umapper.com. Natomiast na stronie www.afcomponents.com/tutorials/umap_as3 dostępne są tutoriale dotyczące tego API.

Obliczanie odległości między punktami

Jeśli znamy współrzędne geograficzne dwóch różnych punktów, możemy określić odległość między nimi. AIR nie udostępnia żadnego API, którym można by się posłużyć, jednak możesz skorzystać z rozwiązania AS3 opisanego w serwisie Adobe Developer Connection, na stronie <http://cookbooks.adobe.com/index.cfm?event=showdetails&postId=5701>.

Odległość można również wyznaczyć, korzystając z API UMap ActionScript 3.0. Sprawdź to na stronie www.umapper.com.

Geokodowanie

Dostawcy map, na przykład Google czy Yahoo!, udostępniają internetowe usługi geokodowania oraz odwrotnego geokodowania. Geokodowanie polega na znalezieniu szerokości i długości geograficznej adresu, natomiast odwrotne geokodowanie pozwala uzyskać adres punktu na podstawie pary współrzędnych geograficznych.

Z poziomu aplikacji AIR for iOS możesz wysłać zapytanie HTTP do jednego z tych serwisów. Przykładem może być serwis internetowy Yahoo! PlaceFinder, dostępny pod adresem <http://developer.yahoo.com/geo/placefinder>.

Inną metodą jest zintegrowanie API UMap ActionScript 3.0 z jednym z tych serwisów, co pozwoli na włączenie funkcji geokodowania bezpośrednio w projektach programu Flash. Więcej informacji na ten temat znajdziesz na stronie uMapper.

Korzystanie z żyroskopu

Kolejnym popularnym czujnikiem jest żyroskop, który można znaleźć w większości najnowszych urządzeń iOS. AIR SDK nie obsługuje żyroskopu bezpośrednio, jednak firma Adobe przygotowała natywne rozszerzenie środowiska AIR 3.0, które zawiera klasę ActionScript o nazwie Gyroscope.

Rozszerzenie to można pobrać z serwisu Adobe Developer Connection wraz z przykładami użycia. Sprawdź na stronie www.adobe.com/devnet/air/native-extensions-for-air/extensions/gyroscope.html.

Zobacz też

- „Określanie prędkości i kierunku poruszania się”.
- „Uruchamianie aplikacji Mapy” w rozdziale 13.

Określanie prędkości i kierunku poruszania się

Dostępność odbiornika GPS na urządzeniu pozwala na określenie prędkości oraz kierunku poruszania się. W tej recepturze napiszemy prostą aplikację, która korzysta z klasy `Geolocation` do uzyskania i wykorzystania tych informacji. Aplikację wyposażymy w kompas, który będzie działał na podstawie kierunku poruszania się użytkownika.

Przygotowanie

Potrzebne nam będzie urządzenie iOS z odbiornikiem GPS. Można je znaleźć w iPhone'ach, począwszy od wersji 3G. Odbiorniki GPS są również instalowane na iPadach działających w sieci komórkowej.

W programie Flash Professional otwórz plik *recipe fla* znajdujący się w katalogu *rozdzial09\receptura2* w pakiecie dołączonym do książki.

Na stole montażowym znajdują się trzy dynamiczne pola tekstowe. Pierwsze dwa (`speed1Field` i `speed2Field`) posłużą do wyświetlenia bieżącej prędkości odpowiednio w metrach na sekundę oraz w milach na godzinę. W trzecim polu `headingField` wyświetlony zostanie kierunek poruszania się.

W dolnej części stołu montażowego umieszczono klip filmowy compass, który pokazuje kompas z zaznaczonymi kierunkami północnym, południowym, wschodnim oraz zachodnim. W odpowiedzi na zmiany kierunku poruszania się będziemy odświeżać obrót tego klipu filmowego tak, aby zawsze wskazywał północ.

Jak to zrobić...

Wykonaj poniższe czynności, aby odczytać prędkość i kierunek poruszania się urządzenia:

1. Utwórz klasę dokumentu o nazwie Main.
2. Dodaj niezbędne instrukcje importu, utwórz stałą oraz zmienną prywatną typu `Geolocation`:

```
package {
    import flash.display.MovieClip;
    import flash.events.GeolocationEvent;
    import flash.sensors.Geolocation;
    public class Main extends MovieClip {
        private const CONVERSION_FACTOR:Number = 2.237;
        private var geo:Geolocation;
        public function Main() {
            // constructor code
        }
    }
}
```

3. W konstruktorze zainicjalizuj obiekt `Geolocation` i zacznij nasłuchiwać na jego aktualizacje:

```
public function Main() {
    if(Geolocation.isSupported)
    {
        geo = new Geolocation();
        geo.setRequestedUpdateInterval(50);
        geo.addEventListener(GeolocationEvent.UPDATE, geoUpdated);
    }
}
```

4. Będzie nam potrzebna funkcja nasłuchująca na zdarzenie `UPDATE` wywoływane przez obiekt `Geolocation`. W funkcji tej będziemy uzyskiwać i wyświetlać bieżącą prędkość oraz kierunek, a także uaktualniać klip filmowy compass, aby zawsze wskazywał kierunek północny. Napisz następującą metodę:

```
private function geoUpdated(e:GeolocationEvent):void {
    var metersPerSecond:Number = e.speed;
    var milesPerHour:uint = getMilesPerHour(metersPerSecond);
    speed1Field.text = String(metersPerSecond);
    speed2Field.text = String(milesPerHour);
    var heading:Number = e.heading;
```



```
compass.rotation = 360 - heading;
headingField.text = String(heading);
}
```

5. Następnie dodaj pomocniczą metodę służącą do przeliczenia prędkości w metrach na sekundę na prędkość w milach na godzinę:

```
private function getMilesPerHour(metersPerSecond:Number):uint
{
    return metersPerSecond * CONVERSION_FACTOR;
}
```

6. Zapisz klasę w pliku *Main.as*. Powróć do pliku FLA i zapisz również projekt.
7. Skompiluj projekt FLA i zainstaluj plik IPA na urządzeniu.
8. Uruchom aplikację. Jeśli to konieczne, udostępnij aplikacji dostęp do jednostki GPS.

Trzymaj urządzenie przed sobą i zacznij je obracać. Pole **heading (degrees)** będzie się odświeżać, pokazując kierunek, w jakim zwrócone jest urządzenie. Uaktualniony zostanie także klip filmowy *compass*, który będzie pokazywać, gdzie znajduje się północ.

Wybierz się na spacer i zabierz ze sobą urządzenie. Zaczynj chodzić, a najlepiej pobiegij. Średnio co 50 milisekund dwa górne pola tekstowe będą się odświeżać, pokazując bieżącą prędkość zarówno w metrach na sekundę, jak i w milach na godzinę.

Opis działania

W tym ćwiczeniu utworzyłeś obiekt typu `Geolocation` i nasłuchiwałeś na wywoływane przez niego zdarzenia `UPDATE`. Zdefiniowałeś interwał odświeżania równy 50 milisekund. W takich odstępach czasu będą podejmowane próby odczytania prędkości oraz kierunku.

Zarówno prędkość, jak i kierunek są odczytywane z obiektu `GeolocationEvent`, który jest wywoływany przez każde zdarzenie `UPDATE`. Obiekt zdarzenia jest przechwytywany i obsługiwany przez funkcję `geoUpdated()`, która odpowiada za wyświetlenie prędkości i informacji o kierunku uzyskanych z akcelerometru.

Bieżąca prędkość jest mierzona w metrach na sekundę i można ją uzyskać poprzez sprawdzenie właściwości `GeolocationEvent.speed`. Nasza funkcja obsługi zdarzenia przed wyświetleniem każdej wartości w odpowiednim polu tekstowym przelicza prędkość w metrach na sekundę na mile na godzinę. Odpowiada za to następujący kod:

```
var metersPerSecond:Number = e.speed;
var milesPerHour:uint = getMilesPerHour(metersPerSecond);
speed1Field.text = String(metersPerSecond);
speed2Field.text = String(milesPerHour);
```

Kierunek, w którym zwrócone jest urządzenie, a który odpowiada kierunkowi poruszania się (w odniesieniu do północy), jest odczytywany z właściwości `GeolocationEvent.heading`. Jej

wartość służy do ustawienia wartości właściwości `rotation` klipu filmowego `compass`, która jest wyświetlana w polu tekstowym `headingField`:

```
var heading:Number = e.heading;
compass.rotation = 360 - heading;
headingField.text = String(heading);
```

Ostatnia metoda to `getMilesPerHour()`, która jest wykorzystywana w funkcji `geoUpdated()` do przeliczenia bieżącej prędkości w metrach na sekundę na prędkość w milach na godzinę. Zwróć uwagę na użycie stałej `CONVERSION_FACTOR`, która została zadeklarowana w klasie dokumentu:

```
private function getMilesPerHour(metersPerSecond:Number):uint
{
    return metersPerSecond * CONVERSION_FACTOR;
}
```

Mimo że prędkość i kierunek, odczytane na podstawie danych zarejestrowanych przez odbiornik GPS, wystarczą dla większości aplikacji, to ich dokładność może być różna na innych urządzeniach. Wpływ na odczyt ma również otoczenie; przemieszczanie się po ulicach z wysokimi budynkami lub między drzewami może doprowadzić do zakłóceń odczytu.

Więcej informacji na temat klas `flash.sensors.Geolocation` i `flash.events.GeolocationEvent` znajdziesz w systemie pomocy Adobe Community Help.

Dodatkowe informacje

Poniżej opisano kilka dodatkowych zagadnień.

Sprawdzanie dostępności odbiornika

Bieżącą prędkość i kierunek poruszania się użytkownika można określić jedynie na urządzeniach wyposażonych w odbiornik GPS.

Mimo że aplikację utworzoną w tym ćwiczeniu można uruchomić na dowolnym urządzeniu iOS, to w przypadku dowolnego modelu urządzeń iPod touch, oryginalnego iPhone'a czy iPadów obsługujących jedynie sieć Wi-Fi nie uzyskasz żadnych odczytów. Wartość właściwości `GeolocationEvent.speed` będzie wynosić `-1`, zaś właściwość `GeolocationEvent.heading` zwróci wartość `NaN`.

Jeśli Twoja aplikacja wymaga obecności odbiornika GPS, należy umieścić odpowiedni wpis w pliku deskryptora aplikacji. Dzięki temu użytkownicy, których urządzenia nie spełniają tego wymagania, nie będą pobierać Twojej aplikacji z App Store.

Więcej informacji na ten temat znajdziesz w recepturze „Określanie wymagań urządzenia” w rozdziale 13.

Symulowanie odbiornika GPS

Podczas rozwijania aplikacji ciągle testowanie na urządzeniu może być uciążliwe. Możesz tego uniknąć, zapisując dane odczytane przez urządzenie i korzystając z nich podczas testowania. Do dyspozycji masz szereg aplikacji, które pozwolą na zapisanie danych z czujników dostępnych na urządzeniu.

Jedną z takich aplikacji jest xSensor, który można pobrać za darmo z iTunes lub z App Store. Darmowa wersja pozwala na zapisanie jedynie 50 kB danych, jednak możesz zakupić xSensor Pro, który jest pozbawiony tego ograniczenia.

Zapobieganie przejściu w stan uśpienia

Wiele aplikacji utworzonych w tym rozdziale nie wymaga zbyt częstej interakcji dotykowej. Może to spowodować przygaszenie ekranu lub jego zablokowanie podczas testowania. Jest to dość uciążliwe i można temu zapobiec poprzez wyłączenie blokowania ekranu. Więcej informacji na ten temat znajdziesz w recepturze „Zapobieganie przejściu ekranu w stan bezczynności” w rozdziale 3.

Zobacz też

- „Określanie bieżącego położenia”.
- „Sprawdzanie dostępności geolokalizacji”.

Sprawdzanie dostępności geolokalizacji

Aplikacje, które korzystają z danych o położeniu urządzenia, muszą otrzymać odpowiednie pozwolenie użytkownika. Gdy aplikacja po raz pierwszy podejmie próbę uzyskania dostępu do danych o położeniu, na ekranie pojawi się komunikat informujący o konieczności wydania pozwolenia. Urządzenia, na których zainstalowano iOS w wersji 4 lub wyższej, będą pamiętać ten wybór; natomiast starsze wersje iOS będą wymagać zgody na dostęp podczas każdego uruchomienia aplikacji. Ponadto uprawnienia dostępu do informacji o lokalizacji można zmienić w dowolnym momencie, korzystając z ustawień urządzenia.

Bardzo ważne jest, aby aplikacja wykrywała dostępność danych geolokalizacyjnych i reagowała na zmianę uprawnień w trakcie działania programu. Sprawdźmy, jak to zrobić.

Przygotowanie

Na potrzeby tej receptury przygotowano wstępny projekt.

W programie Flash Professional otwórz plik *recipe.fla* znajdujący się w katalogu *rozdzial09\receptura3* w pakiecie dołączonym do książki.

Na stole montażowym umieszczono dynamiczne pole tekstowe, którego instancja nosi nazwę *output*.

Napišemy aplikację, która będzie wykrywać dostępność danych geolokalizacyjnych oraz raportować wszelkie zmiany w polu tekstowym *output*.

Jak to zrobić...

Wykonaj opisane poniżej czynności:

1. Utwórz klasę dokumentu i nazwij ją *Main*.
2. Dodaj następujące instrukcje importu i zadeklaruj zmienną prywatną typu *Geolocation*:

```
package {
    import flash.display.MovieClip;
    import flash.events.GeolocationEvent;
    import flash.events.StatusEvent;
    import flash.sensors.Geolocation;
    public class Main extends MovieClip {
        private var geo:Geolocation;
        public function Main() {
            // constructor code
        }
    }
}
```

3. W konstruktorze utwórz obiekt *Geolocation* i rozpocznij nasłuchiwanie na wywoływane przez niego zdarzenia *GeolocationEvent.UPDATE* i *StatusEvent.STATUS*:

```
public function Main() {
    output.text = "Sprawdzanie położenia...";
    if(Geolocation.isSupported)
    {
        geo = new Geolocation();
        geo.setRequestedUpdateInterval(1000);
        geo.addEventListener(GeolocationEvent.UPDATE, geoUpdated);
        geo.addEventListener(StatusEvent.STATUS, statusUpdated);
    }
    else
    {
        output.text = "Geolokalizacja nie jest dostępna.";
    }
}
```

4. Następnie utwórz funkcję obsługi dla każdego zdarzenia:

```
private function geoUpdated(e:GeolocationEvent):void {
    output.text = "Uzyskano położenie.";
}
private function statusUpdated(e:StatusEvent):void {
    if(e.code == "Geolocation.Muted")
    {
        output.text = "Odmowa dostępu do geolokalizacji.";
    }
}
```

5. Zapisz plik klasy pod nazwą `Main.as`.

6. Zapisz projekt FLA i opublikuj go. Zainstaluj plik IPA na urządzeniu i uruchom aplikację.

Na ekranie pojawi się natywne okno dialogowe iOS z następującym ostrzeżeniem:

„c9 r3” chce użyć informacji o Twoim położeniu.

7. Stuknij przycisk **Nie pozwalaj**, aby uniemożliwić dostęp aplikacji do danych lokalizacyjnych.

Na ekranie pojawi się następujący komunikat:

Sprawdzanie położenia...

Zostanie on szybko zastąpiony przez:

Odmowa dostępu do geolokalizacji.

8. Następnie uruchom aplikację ponownie. Jeśli używasz iOS w wersji 4 lub wyższej, poprzednio wybrane ustawienia zostaną zapamiętane i dostęp do danych geolokalizacyjnych zostanie zablokowany. Natomiast w przypadku wcześniejszych wersji iOS podczas każdego uruchamiania aplikacji będziesz musiał udzielić aplikacji odpowiedniego zezwolenia.

Aby ponownie uruchomić aplikację w środowisku iOS w wersji 4 lub wyższej, musisz ją najpierw zamknąć z użyciem szybkiego przełącznika aplikacji. Zerknij do receptury „Eleganckie zamykanie aplikacji” w rozdziale 3, aby sprawdzić, jak to zrobić.

Na urządzeniach, na których zainstalowano iOS w wersji 4 lub wyższej, uprawnienia dostępu aplikacji można zmienić poprzez zmianę ustawień urządzenia. Zróbmy to teraz dla naszej aplikacji.

1. Wyjdź z aplikacji, naciskając przycisk *Home*. Przejdź do ustawień urządzenia i wyświetl sekcję **Usługi lokalizacji**. Do dyspozycji będziesz mieć listę aplikacji, które próbowały uzyskać dostęp do danych lokalizacyjnych. Przewiń listę w dół, aż ujrzysz aplikację **c9 r3**. Stuknij przycisk obok niej, aby nadać jej uprawnienia dostępu.



2. Następnie powróć do ekranu głównego i ponownie uruchom aplikację z tego ćwiczenia.

Tym razem na ekranie pojawi się następujący napis:

Sprawdzanie położenia...

Zostanie on szybko zastąpiony przez:

Uzyskano położenie.

Opis działania

Gdy użytkownik nie zezwoli aplikacji AIR for iOS na dostęp do danych lokalizacyjnych, zdarzenia `GeolocationEvent.UPDATE` wygasną, zaś obiekt `Geolocation` wyzwoi zdarzenie `StatusEvent.STATUS`. Dzięki temu można wykryć przywrócenie prawa dostępu do danych lokalizacyjnych poprzez nasłuchiwanie na zdarzenie `STATUS`:

```
geo = new Geolocation();
geo.setRequestedUpdateInterval(1000);
geo.addEventListener(GeolocationEvent.UPDATE, geoUpdated);
geo.addEventListener(StatusEvent.STATUS, statusUpdated);
```

Ostateczne sprawdzenie odbywa się w funkcji `statusUpdated()`:

```
private function statusUpdated(e:StatusEvent):void {
    if(e.code == "Geolocation.Muted")
    {
        output.text = "Odmowa dostępu do geolokalizacji.";
    }
}
```

W tej metodzie sprawdzamy wartość właściwości `code` obiektu `StatusEvent`. Jeśli zawiera ona łańcuch `Geolocation.Muted`, wiadomo, że nie mamy dostępu do danych geolokalizacyjnych.

Co prawda nie korzystaliśmy z tego w niniejszej recepturze, ale dostęp do danych geolokalizacyjnych możesz też sprawdzić na podstawie właściwości `Geolocation.muted`. Gdy nowo zainstalowana aplikacja zostanie uruchomiona po raz pierwszy, właściwość `muted` będzie miała wartość `true` aż do chwili, gdy użytkownik nada aplikacji prawa dostępu w natywnym oknie dialogowym iOS.

Więcej informacji na temat `flash.events.StatusEvent` można znaleźć w systemie pomocy Adobe Community Help.

Zobacz też

- „Określanie bieżącego położenia”.

Reagowanie na zmiany zarejestrowane przez akcelerometr

Akcelerometr pozwala uzyskać dane określające położenie lub przesunięcie urządzenia względem trójwymiarowych osi. Zarejestrowane dane można odczytać z użyciem kodu `ActionScript`.

Z tej receptury dowiesz się, jak skorzystać z możliwości akcelerometru wbudowanego w urządzeniu iOS.

Przygotowanie

Na potrzeby tej receptury przygotowano wstępny projekt.

W programie Flash Professional otwórz plik *recipe fla* znajdujący się w katalogu *rozdzial09\receptura4* w pakiecie dołączonym do książki.

Na stole montażowym umieszczono pięć dynamicznych pól tekstowych. Poniżej znajduje się klip filmowy, którego instancja nosi nazwę `arrow`. W każdym polu tekstowym będziemy wyświetlać dane odczytane z akcelerometru urządzenia. Będziemy też obracać klip filmowy, aby odzwierciedlić fizyczne zmiany w położeniu urządzenia.

Zwróć uwagę na wymiary stołu montażowego, które zmieniono na 480×320. W tej recepturze będziemy korzystać z orientacji poziomej.

Jak to zrobić...

Wykonaj poniższe czynności, aby wykryć i zareagować na zmiany odczytów akcelerometru:

1. Utwórz klasę dokumentu i nazwij ją Main.
2. Dodaj następujące dwie instrukcje importu i zadeklaruj zmienną prywatną typu Accelerometer:

```
package {
    import flash.display.MovieClip;
    import flash.events.AccelerometerEvent;
    import flash.sensors.Accelerometer;
    public class Main extends MovieClip {
        private var acc:Accelerometer;
        public function Main() {
            // constructor code
        }
    }
}
```

3. W konstruktorze zainicjalizuj obiekt Accelerometer i rozpocznij nasłuchiwanie na jego aktualizacje:

```
public function Main() {
    if(Accelerometer.isSupported)
    {
        acc = new Accelerometer();
        acc.setRequestedUpdateInterval(50);
        acc.addEventListener(AccelerometerEvent.UPDATE,
            accUpdated);
    }
}
```

4. Zakończ, tworząc funkcję obsługi zdarzenia, która będzie pobierać i wykorzystywać uaktualnione dane odczytane z akcelerometru:

```
private function accUpdated(e:AccelerometerEvent):void {
    var radians:Number = Math.atan2(e.accelerationY,
        e.accelerationX);
    var degrees:Number = (radians * (180 / Math.PI)) - 90;
    arrow.rotation = -degrees;
    accXField.text = e.accelerationX.toString();
    accYField.text = e.accelerationY.toString();
    accZField.text = e.accelerationZ.toString();
    timeField.text = e.timestamp.toString();
    rotField.text = degrees.toString();
}
```

5. Zapisz plik klasy pod nazwą Main.as.
6. Zapisz projekt FLA i opublikuj go. Zainstaluj aplikację na urządzeniu i uruchom ją.

Gdy tylko czujnik ruchu urządzenia wykryje przesunięcie, zawartość pól tekstowych zostanie odświeżona. Trzymaj urządzenie przed sobą i przechylaj je zgodnie z ruchem wskazówek zegara lub w przeciwnym kierunku, aby uaktualnić wartość obrotu klipu filmowego arrow, co gwarantuje, że strzałka będzie zawsze skierowana w górę.

Opis działania

W tym ćwiczeniu skorzystaliśmy z danych akcelerometru, tworząc instancję klasy `Accelerometer` i nasłuchując na wyzwalane przez nią zdarzenie `AccelerometerEvent.UPDATE`:

```
acc = new Accelerometer();
acc.setRequestedUpdateInterval(50);
acc.addListener(AccelerometerEvent.UPDATE, accUpdated);
```

Częstotliwość, z jaką odbierane są zdarzenia `UPDATE`, można określić, wywołując metodę `Accelerometer.setRequestedUpdateInterval()`. Widać to w powyższym fragmencie kodu, w którym zdefiniowaliśmy odświeżanie w odstępach 50-milisekundowych. Jest to jedynie wskazówka dla urządzenia, ponieważ rzeczywisty czas między kolejnymi odczytami może być nieco krótszy lub nieco dłuższy. Pominięcie wywołania tej metody spowoduje użycie domyślnego odstępu czasu, który może mieć różną wartość: od kilku milisekund aż do kilku sekund, w zależności od możliwości sprzętowych urządzenia.

Zdarzenie `UPDATE` to obiekt typu `AccelerometerEvent`, który ma następujące właściwości:

- `accelerationX` — przyspieszenie wzdłuż osi x . Gdy urządzenie znajduje się w pozycji pionowej, wówczas oś x przebiega od lewej do prawej. Przyspieszenie jest dodatnie, gdy urządzenie przemieszcza się w prawo.
- `accelerationY` — przyspieszenie wzdłuż osi y . Gdy urządzenie znajduje się w pozycji pionowej, wówczas oś y przebiega z dołu do góry. Przyspieszenie jest dodatnie, gdy urządzenie przemieszcza się w górę.
- `accelerationZ` — przyspieszenie wzdłuż osi z . Przyspieszenie jest dodatnie, gdy urządzenie jest przesuwane w taki sposób, że jego ekran jest skierowany w górę. Przyspieszenie jest ujemne, jeśli ekran jest skierowany w dół.
- `timestamp` — liczba milisekund, które upłynęły od ostatniego uruchomienia aplikacji.

Przyspieszenie jest mierzone w odniesieniu do standardowego przyspieszenia ziemskiego, będącego wynikiem działania grawitacji. Jego wartość wynosi około 9,8 metrów na sekundę.

Właściwości te odczytaliśmy w funkcji `accUpdated()`, która odpowiada również za umieszczenie ich wartości w dynamicznych polach tekstowych:

```
accXField.text = e.accelerationX.toString();
accYField.text = e.accelerationY.toString();
accZField.text = e.accelerationZ.toString();
timeField.text = e.timestamp.toString();
```

Dodatkowo korzystamy z właściwości `accelerometerX` i `accelerometerY`, które służą do obliczenia kąta nachylenia urządzenia (z ekranem zwróconym w Twoją stronę). Wielkość tego kąta służy do odświeżenia obrotu klipu filmowego arrow. Odpowiada za to następujący fragment kodu:

```
var radians:Number = Math.atan2(e.accelerationY,  
    e.accelerationX);  
var degrees:Number = (radians * (180 / Math.PI)) - 90;  
arrow.rotation = -degrees;
```

Znajomość kąta, pod jakim nachylone zostało urządzenie, można wykorzystać w wielu aplikacjach, a w szczególności w niektórych grach. Nachylenie urządzenia może posłużyć do przemieszczania bohaterów na ekranie lub do symulowania ruchu kierownicy.

Zwróć uwagę na użycie w konstruktorze statycznej właściwości `Accelerometer.isSupported`. Właściwość ta jest tylko do odczytu i pozwala na sprawdzenie dostępności akcelerometru. Czujnik ten jest wbudowany we wszystkie dostępne obecnie urządzenia iOS, jednak nie można przewidzieć, co się stanie w przyszłości. Dlatego dobrze jest sprawdzić jego dostępność, również podczas rozwijania aplikacji na różne platformy.

Dodatkowe informacje

Poniżej opisano kilka dodatkowych zagadnień związanych z akcelerometrem i wykorzystaniem uzyskanych danych.

Orientacja i osie akcelerometru

Osie akcelerometru są skorelowane z ekranem, a nie z fizyczną orientacją samego urządzenia. Innymi słowy, gdy włączona jest automatyczna orientacja i gdy urządzenie jest ustawione pionowo, oś *y* ma kierunek pionowy, zaś oś *x* — poziomy. Stwierdzenie to jest prawdziwe dla aplikacji, które są domyślnie wyświetlane w orientacji pionowej lub poziomej. Natomiast jeśli automatyczna orientacja nie jest włączona, osie akcelerometru nie zostaną dopasowane do nowego położenia po obróceniu urządzenia.

Określanie orientacji urządzenia

Odczyty akcelerometru podlegają wpływowi grawitacji i można ich używać do określenia bieżącej orientacji urządzenia. Poniżej wypisano wartości, które należy sprawdzić:

- `accelerationX > 0.5` — obrót o 90° w kierunku przeciwnym do ruchu wskazówek zegara.
- `accelerationX < -0.5` — obrót o 90° w kierunku zgodnym z ruchem wskazówek zegara.
- `accelerationY > 0.5` — normalne położenie w pionie.
- `accelerationY < -0.5` — położenie do góry nogami.
- `accelerationZ > 0.5` — ekran skierowany w górę.
- `accelerationZ < -0.5` — ekran skierowany w dół.

Jest to alternatywny sposób na określenie orientacji, którego można użyć zamiast nasłuchiwanie na obiekty `StageOrientationEvent` wyzwalane przez stół montażowy. Dodatkowo używanie danych akcelerometru pozwala określić, czy ekran urządzenia jest skierowany w górę, czy w dół.

Stosowanie filtra dolnoprzepustowego

Na dane z akcelerometru ma wpływ zarówno grawitacja, jak i nagle zmiana ruchu. Jeśli korzystasz z tych danych do wykrywania orientacji urządzenia, powinieneś wyizolować z nich składową grawitacyjną poprzez zastosowanie filtra dolnoprzepustowego.

Można to osiągnąć, uśredniając dane wraz z upływem czasu. W tym celu należy zdefiniować współczynnik filtrowania oraz trzy zmienne prywatne, które przechowują poprzednie wartości dla każdej osi:

```
private const FACTOR:Number = 0.1;
private var accX:Number = 0;
private var accY:Number = 0;
private var accZ:Number = 0;
```

Następnie w odpowiedzi na każde zdarzenie `AccelerometerEvent.UPDATE` stosujemy filtr dolnoprzepustowy, który pozwoli wyodrębnić dla każdej osi jedynie komponent grawitacji:

```
accX = (e.accelerationX * FACTOR) + (accX * (1 - FACTOR));
accY = (e.accelerationY * FACTOR) + (accY * (1 - FACTOR));
accZ = (e.accelerationZ * FACTOR) + (accZ * (1 - FACTOR));
```

Powyższy kod służy do obliczenia wartości dla każdej osi na podstawie 10 procent bieżącej wartości oraz 90 procent wartości obliczonej wcześniej. Gwarantuje to, że wartości odczytów będą się zmieniać powoli w odpowiedzi na nagle i krótkie zmiany ruchu.

Stosowanie filtra górnoprzepustowego

Wiele rodzajów aplikacji korzysta z danych akcelerometru, aby wykrywać gwałtowne zmiany ruchu. Filtr górnoprzepustowy może służyć do wyodrębnienia składowej nagłych zmian ruchu.

Podobnie jak w przypadku implementacji filtra dolnoprzepustowego, również teraz zdefiniujemy współczynnik filtrowania oraz trzy zmienne prywatne, które będą przechowywać wcześniejsze wartości dla każdej osi. Następnie w odpowiedzi na każde zdarzenie `AccelerometerEvent.UPDATE` zastosujemy filtr:

```
accX = e.accelerationX - ((e.accelerationX * FACTOR) +
    (accX * (1 - FACTOR)));
accY = e.accelerationY - ((e.accelerationY * FACTOR) +
    (accY * (1 - FACTOR)));
accZ = e.accelerationZ - ((e.accelerationZ * FACTOR) +
    (accZ * (1 - FACTOR)));
```

W tym przykładzie dla każdej osi obliczamy wartość filtra dolnoprzepustowego, którą następnie odejmujemy od bieżącej wartości osi. Dzięki temu po wyeliminowaniu czynnika grawitacyjnego możemy reagować na nagłe zmiany ruchu.

Właściwość muted

W klasie Accelerometer zdefiniowano statyczną właściwość muted, która jest tylko do odczytu. Służy ona do sprawdzenia, czy użytkownik zezwolił aplikacji na dostęp do danych zarejestrowanych przez akcelerometr. Właściwość ta nie jest wymagana przez iOS, ponieważ obecnie nie można odmówić żadnej aplikacji dostępu do akcelerometru.

Zobacz też

- „Ustawianie domyślnego formatu ekranu” w rozdziale 8.
- „Włączanie automatycznej orientacji” w rozdziale 8.
- „Reakcja na zmiany orientacji” w rozdziale 8.

Wykrywanie potrząśnięcia

Akcelerometr często służy do wykrywania potrząśnięcia, które stało się popularną metodą interakcji w grach i aplikacjach. Przykładem są preinstalowane aplikacje iOS, w których można cofnąć zmiany poprzez potrząśnięcie urządzeniem.

W tej recepturze pokażemy, jak wykryć potrząśnięcie urządzeniem na podstawie danych zarejestrowanych przez akcelerometr.

Przygotowanie

W programie Flash Professional otwórz plik *recipe fla* znajdujący się w katalogu *rozdzial09\receptura5* w pakiecie dołączonym do książki.

Pośrodku sceny znajduje się klip filmowy o nazwie shake. Jego oś czasu składa się z dwóch klatek.

Napiszemy kod ActionScript, który będzie przesuwiał klip w odpowiedzi na ruch urządzenia względem trzech osi. Gdy urządzenie rejestruje ruch, aplikacja przejdzie do drugiej klatki klipu, aby poinformować użytkownika o wykryciu potrząśnięcia.

Stół montażowy w tym projekcie jest zorientowany poziomo.

Jak to zrobić...

Wykonaj poniższe czynności, aby wykryć potrząśnięcie:

1. Utwórz klasę dokumentu i nazwij ją Main.
2. Zaimportuj klasy niezbędne do obsługi akcelerometru i utwórz następujące zmienne prywatne:

```
package {
    import flash.display.MovieClip;
    import flash.events.AccelerometerEvent;
    import flash.sensors.Accelerometer;
    public class Main extends MovieClip {
        private const THRESHOLD:Number = 0.8;
        private var acc:Accelerometer;
        private var prevX:Number;
        private var prevY:Number;
        private var prevZ:Number;
        private var startX:Number;
        private var startY:Number;
        public function Main() {
            // constructor code
        }
    }
}
```

3. Zainicjalizuj zmienne prywatne i klip filmowy shake. Ponadto zainicjalizuj obiekt Accelerometer i rozpocznij nasłuchiwanie na wyzwalane przezeń zdarzenia UPDATE:

```
public function Main() {
    prevX = prevY = prevZ = 0;
    shake.gotoAndStop(1);
    startX = shake.x;
    startY = shake.y;
    if(Accelerometer.isSupported)
    {
        acc = new Accelerometer();
        acc.setRequestedUpdateInterval(50);
        acc.addEventListener(AccelerometerEvent.UPDATE,
            accUpdated);
    }
}
```

4. Dodaj funkcję obsługi zdarzenia UPDATE:

```
private function accUpdated(e:AccelerometerEvent):void {
    var changeX:Number = prevX - e.accelerationX;
    var changeY:Number = prevY - e.accelerationY;
    var changeZ:Number = prevZ - e.accelerationZ;
    prevX = e.accelerationX;
    prevY = e.accelerationY;
```

```

prevZ = e.accelerationZ;
shake.x = startX + (changeX * 100);
shake.y = startY + (changeY * 100);
shake.z = (changeZ * 100);
if(Math.abs(changeX) > THRESHOLD ||
    Math.abs(changeY) > THRESHOLD ||
    Math.abs(changeZ) > THRESHOLD)
{
    shake.gotoAndStop(2);
}
else
{
    shake.gotoAndStop(1);
}
}

```

5. Zapisz plik klasy pod nazwą `Main.as`.

6. Zapisz projekt FLA i opublikuj go. Przetestuj aplikację na urządzeniu.

Po delikatnym potrząśnięciu urządzeniem na środku ekranu wyświetli się klip filmowy. Gwałtowniejsze potrząśnięcie spowoduje zmianę jego wyglądu, co będzie oznaczać wykrycie wystarczająco mocnego potrząśnięcia.

Opis działania

W tym przykładzie porównujemy bieżący odczyt akcelerometru z poprzednim, co pozwala stwierdzić, czy zaszła wystarczająco duża zmiana.

Poprzednie dane odczytane z akcelerometru są przechowywane w zmiennych prywatnych `prevX`, `prevY` oraz `prevZ`. Wartości te są porównywane z bieżącymi danymi, aby określić zmiany dla każdej osi:

```

var changeX:Number = prevX - e.accelerationX;
var changeY:Number = prevY - e.accelerationY;
var changeZ:Number = prevZ - e.accelerationZ;
prevX = e.accelerationX;
prevY = e.accelerationY;
prevZ = e.accelerationZ;

```

Jeśli zmiana dla dowolnej osi jest wystarczająco duża, możemy przypuszczać, że użytkownik potrząsnął urządzeniem. Ta informacja jest przekazywana użytkownikowi poprzez wyświetlenie drugiej klatki klipu filmowego `shake`. Odpowiada za to następujący fragment kodu:

```

if(Math.abs(changeX) > THRESHOLD ||
    Math.abs(changeY) > THRESHOLD ||
    Math.abs(changeZ) > THRESHOLD)
{
    shake.gotoAndStop(2);
}

```

```

}
else
{
    shake.gotoAndStop(1);
}

```

Stała `THRESHOLD` określa wielkość zmiany wartości dowolnej osi, potrzebnej do stwierdzenia, że dany ruch to potrząsanie. Po obniżeniu tej wartości wystarczy delikatniejszy ruch, aby wykryto potrząśnięcie, natomiast po jej zwiększeniu użytkownik będzie musiał mocniej potrząsnąć urządzeniem.

Uaktualniamy też współrzędne x , y i z klipu filmowego `shake` w odpowiedzi na zmiany odczytu akcelerometru, aby wzmocnić efekt wizualny. Im gwałtowniejsze potrząsanie, tym bardziej odsuwamy klip od początkowego położenia:

```

shake.x = startX + (changeX * 100);
shake.y = startY + (changeY * 100);
shake.z = (changeZ * 100);

```

Zmienne prywatne `startX` i `startY` są inicjalizowane w konstruktorze wartościami początkowych współrzędnych klipu filmowego `shake`.

Większość zadań w tej recepturze jest wykonywana w funkcji obsługi zdarzenia `accUpdated()`, która jest wywoływana po każdym przechwyceniu zdarzenia `AccelerometerEvent.UPDATE` wyzwanego przez obiekt `Accelerometer`. W funkcji tej definiujemy częste odświeżanie odczytów, wywołując metodę `setRequestedUpdateInterval()`, aby zagwarantować szybką reakcję aplikacji na zmiany.

Dodatkowe informacje

Przyjrzyjmy się jeszcze kilku dodatkowym opcjom, które pozwolą na ulepszenie przykładowej aplikacji.

Sprawdzanie wielu osi

Dotychczas do wykrycia potrząsania wystarczyła duża zmiana wartości dla jednej z osi. Inne podejście polega na oczekiwaniu na znaczącą zmianę dla dwóch lub trzech osi. Potrzebny kod wygląda następująco:

```

var changeX:Number = Math.abs(e.accelerationX);
var changeY:Number = Math.abs(e.accelerationY);
var changeZ:Number = Math.abs(e.accelerationZ);
if((changeX > THRESHOLD && changeY > THRESHOLD) ||
    (changeX > THRESHOLD && changeZ > THRESHOLD) ||
    (changeY > THRESHOLD && changeZ > THRESHOLD))
{
    // Wykryte potrząsanie.
}

```

Dla wielu osi można zmniejszyć nieco wartość stałej THRESHOLD, aby skompensować konieczność mocniejszego ruchu, niezbędnego do zainicjalizowania potrząśnięcia.

Uśrednianie danych akcelerometru

Być może zauważyłeś, że nawet gdy trzymasz urządzenie nieruchomo, klip filmowy shake lekko drga. Akcelerometry nie są dość dokładne, dlatego ich odczyty są dość zaszumione.

Szumy można zredukować, stosując filtr górnoprzepustowy do kolejnych odczytów. Technika ta została opisana w recepturze „Reagowanie na zmiany zarejestrowane przez akcelerometr”. Spróbuj poeksperymentować ze współczynnikiem filtra i wartością stałej THRESHOLD, aż uzyskasz zadowalające rezultaty.

Zobacz też

- „Reagowanie na zmiany zarejestrowane przez akcelerometr”.

Skorowidz

A

- Accelerometer, 279
- Accelerometer.isSupported, 280
- Accelerometer.setRequestedUpdateInterval(), 279
- AccelerometerEvent, 279
 - accelerationX, 279
 - accelerationY, 279
 - accelerationZ, 279
 - timestamp, 279
- AccelerometerEvent.UPDATE, 279, 281, 285
- accUpdated(), 279, 285
- ActionScript, 62, 80, 430
 - alokowanie pamięci, 440
 - pula obiektów, 443
 - bitmapa, 192, 194
 - animowanie, 205
 - eksport, 193
 - wyświetlanie, 193
 - deklarowanie typów danych, 430, 431
 - dwustanowy przycisk, 113
 - edycja, 82
 - funkcja blokowania ekranu, 83
 - Gyroscope, 269
 - metody wywołania zwrotnego, 446
 - optymalizacja kodu, 430
 - alokowanie pamięci, 440
 - deklarowanie typów danych, 430, 431
 - jawna rezygnacja z typu danych, 432
 - metody wywołania zwrotnego, 446
 - odśmiecianie, 442
 - optymalizacja pętli, 434, 435
 - pomiar wydajności, 436
 - propagacja zdarzeń, 449, 452
 - pula obiektów, 443
 - sygnały AS3, 449
 - usuwanie nieużytków, 440
 - zastępowanie tablic wektorami, 437
 - optymalizacja pętli, 434, 435
 - dostęp do tablic, 436
 - propagacja zdarzeń, 449, 452
 - zapobieganie, 453
 - przepływ zdarzenia, 452
 - tworzenie klasy dokumentu, 80
 - usuwanie nieużytków, 440
 - odśmiecianie, 442
 - zastępowanie tablic wektorami, 437
 - metody zwracające nowy wektor, 440
- activate(), 86, 225
- activityLevel, 308
- addBitmapData(), 290
- addChild(), 302
- addEventListener(), 452
- Adobe AIR, 58
- ADT, 58, 399, 416, 419, 424
 - instalacja, 425
 - kompilacja aplikacji, 418
 - kompilacja w wierszu poleceń, 425
- afterOrientation, 257
- AIR Debug Launcher, 40, 73
 - publikacja pliku FLA, 73
- AIR Development Tool, *Patrz* ADT

- AIR for iOS, 58, 60, 110
 - ActionScript, 430
 - akcelerometr, 263
 - aparat fotograficzny, 287
 - App Store, 389
 - automatyczna orientacja ekranu, 253
 - rodzaje orientacji, 253
 - włączanie, 253
 - włączanie programistyczne, 254
 - czcionki urządzenia, 210
 - korzyści, 211
 - kroje czcionek, 212
 - kroje pisma, 212
 - lista czcionek, 212, 216, 234
 - osadzanie w aplikacji, 214, 215
 - trudności, 211
 - częstotliwość wyświetlania klatek, 63
 - dokument FLA, 60
 - tworzenie, 61
 - dołączanie pliku HTML, 325
 - dynamiczne generowanie strony WWW, 328
 - dźwięk, 338
 - długość, 361
 - dołączanie pliku MP3, 358
 - głośność, 366
 - kontrola odtwarzania, 362, 363
 - metadane, 361
 - monitorowanie postępu czytania, 360
 - odtwarzanie w tle, 361
 - odtwarzanie nagranych dźwięków, 309
 - odtwarzanie osadzonego dźwięku, 355
 - odtwarzanie pliku MP3, 358
 - odtwarzanie pliku od początku, 365
 - osadzanie w bibliotece, 353, 355
 - panoramowanie, 365
 - pliki PCM, 361
 - wznowienie odtwarzania, 365
 - zatrzymanie odtwarzania, 365
 - zapis nagranych dźwięków, 313
 - zwiększanie czasu bufora, 360
 - Flash Builder, 82
 - format domyślny ekranu, 251
 - geokodowanie, 268
 - gesty, 152
 - machnięcie, 156
 - obracanie obiektu, 162
 - zmiana rozmiaru obiektu, 164
 - przewijanie obiektu, 159
 - włączenie obsługi, 154
 - Interpreter Mode, 405, 421
 - klawiatura wirtualna, 217
 - przewijanie podczas aktywacji, 223
 - typy, 220
 - uruchamianie programistyczne, 220
 - automatyczne przewijanie, 224
 - mikrofon, 287
 - Multi-touch, 137
 - natywne rozszerzenia, 412, 415
 - dołączanie SWC, 412
 - Gyroscope, 412
 - kompilacja warunkowa, 415
 - tworzenie pakietu aplikacji, 417
 - odbiornik GPS, 263, 264
 - odtworzenie nagranych dźwięków, 309
 - konwersja częstotliwości próbkowania, 313
 - określanie bieżącego położenia, 264
 - dokładność danych lokalizacyjnych, 267
 - pobranie danych geolokalizacyjnych, 264
 - wyświetlenie danych geolokalizacyjnych, 264, 266
 - zastosowanie danych geolokalizacyjnych, 268
 - pobieranie tekstu, 221, 222
 - przeglądanie historii wyświetlonych stron, 321
 - adres URL bieżącej strony, 324
 - renderowanie grafiki, 168
 - buforowanie bitmap, 181, 183, 185
 - buforowanie z użyciem macierzy, 187, 188, 189
 - komponowanie sceny, 174
 - rasteryzacja, 174
 - renderowanie programistyczne, 171
 - renderowanie w GPU, 172
 - Stage 3D, 181
 - test wydajności, 168, 169
 - tryb GPU-Blend, 174
 - tryby renderowania, 171, 173, 179
 - renderowanie strony WWW, 318
 - dołączanie banerów reklamowych, 320
 - Safari, 316
 - uruchamianie, 316
 - stół montażowy, 62
 - budowa prostej sceny, 64
 - wymiary, 63
 - uaktualnianie pola tekstowego, 229
 - wczytywanie dołączonego pliku HTML, 326
 - wczytywanie lokalnych stron WWW, 326
 - węskiściowe kontrolki tekstowe iOS, 231

- wejściowe pole tekstowe, 217
 - klawiatura wirtualna, 218
 - pobieranie tekstu, 221
 - tworzenie, 217
- wideo, 338
 - blokowanie ekranu, 342
 - dołączanie pliku FLV, 338
 - dołączanie pliku H.264, 343
 - dołączanie wielu plików, 342
 - kodowanie H.264, 348
 - kontrola odtwarzania pliku, 348, 352
 - maksymalizacja wydajności odtwarzania, 342
 - odtwarzanie pliku H.264, 343, 344
 - odtwarzanie pliku FLV, 338, 339
 - ponowne odtwarzanie pliku, 352
 - punkty sygnalizacji, 341
 - wstrzymanie odtwarzania pliku, 353
 - wyznaczanie rozmiarów klipu, 347
 - wznowienie odtwarzania pliku, 352
- wyświetlanie stron WWW, 316
- zmiany orientacji ekranu, 255
 - blokowanie, 261
 - reakcja, 258
 - wykrywanie, 256
- zrzut ekranu strony WWW, 334
- air.net.SocketMonitor, 371
- air.net.URLMonitor, 371
- akcelerometr, 237, 255, 263, 277
 - orientacja urządzenia, 280
 - osie akcelerometru, 280
 - sprawdzanie wielu osi, 285
 - stosowanie filtra dolnoprzepustowego, 281
 - stosowanie filtra górnoprzepustowego, 281
 - uśrednianie danych, 286
 - wykrywanie potrząśnięcia, 282, 283
 - zmiana odczytów, 278
- aparat fotograficzny, 287
 - Exif, 295
 - odczyt obrazu z rolki, 291
 - użycie domyślnej aplikacji, 296
 - nagrywanie filmu, 299
 - obsługa błędów, 298
 - odczyt zarejestrowanych danych, 299
 - wyświetlanie fotografii, 298
 - zapisywanie w rolce aparatu, 299
 - wybór kamery, 303
 - zapisywanie obrazu w rolce, 288
 - nieudany zapis, 290
- aplikacje iOS, 22
 - ActionScript, 80
 - funkcja blokowania ekranu, 83
 - tworzenie klasy dokumentu, 80
 - AIR Debug Launcher, 40
 - AIR for iOS, 58, 110
 - App Store Review Guidelines, 28
 - automatyczna orientacja ekranu, 253
 - rodzaje orientacji, 253
 - włączanie, 253
 - włączanie programistyczne, 254
 - bitmapa, 170
 - certyfikat P12, 38
 - certyfikat programisty, 32
 - debugowanie, 104
 - dołączanie ikon, 99
 - rodzaje ikon, 100
 - dołączanie pliku HTML, 325
 - dołączanie pliku SWC, 368
 - dopasowanie aplikacji do rozdzielczości, 247
 - dostęp do własnych katalogów, 374
 - stałe dostępu, 375
 - tworzenie katalogów, 376
 - dynamiczne generowanie strony WWW, 328
 - Flash Builder, 82
 - Flash Professional, 22
 - debugowanie, 104
 - format domyślny ekranu, 251
 - funkcja blokowania ekranu, 82
 - grafika wektorowa, 170
 - identyfikator aplikacji, 47
 - identyfikator pakunku, 48
 - identyfikator wartości początkowej pakunku, 48
 - określony identyfikator, 47
 - wieloznaczny identyfikator, 47
 - instalacja aplikacji, 75
 - iTunes, 75
 - Interpreter Mode, 405
 - iOS Developer Program, 22
 - iOS Provisioning Portal, 29
 - rejestracja, 23
 - iOS Human Interface Guidelines, 28
 - iOS Provisioning Portal, 29
 - programistyczny profil informacyjny, 50
 - iPhone Configuration Utility, 78
 - klasa dokumentu, 81
 - kompilacja ahead-of-time, 75
 - kompilator LLVM, 75

- aplikacje iOS
 - konwersja aplikacji Flash, 110
 - kompilacja w wierszu poleceń dla Mac OS X, 406
 - kompilacja w wierszu poleceń dla Microsoft Windows, 400
 - Multi-touch, 138
 - obraz rozruchowy, 96, 241
 - dołączanie, 96
 - iPad, 97
 - orientacja, 97
 - pasek statusu, 98
 - wyświetlacz Retina, 97
 - obsługa wszystkich rozdzielczości, 247
 - odczytywanie plików, 381, 383
 - formaty danych, 383
 - monitorowanie postępu, 384
 - odczyt synchroniczny, 384
 - usuwanie plików, 384
 - określanie wymagań sprzętowych, 395
 - pętla uaktualnień, 92
 - plik deskryptora aplikacji, 101, 372
 - Information Property List, 103
 - określanie wymagań sprzętowych, 395
 - UIStatusBarStyle, 103
 - ustawianie stałego połączenia Wi-Fi, 372, 374
 - pliki SWC, 132
 - statyczne dołączanie, 133
 - pliki SWF, 136
 - porządkowanie, 89
 - przeglądanie historii wyświetlonych stron, 321
 - adres URL bieżącej strony, 324
 - rejestracja urządzenia testującego, 43
 - edycja nazw urządzeń, 46
 - limit urządzeń, 46
 - UDID, 43
 - renderowanie strony WWW, 318
 - dołączanie banerów reklamowych, 320
 - sprawdzanie połączenia, 370
 - interwał odpytywania, 372
 - monitorowanie gniazd, 371
 - symulowanie odbiornika GPS, 273
 - TestFlight, 78
 - tryby dostępu do plików, 384
 - uruchamianie aplikacji Mapy, 391
 - lista obsługiwanych parametrów, 393
 - wyświetlanie aktualnego położenia, 394
 - wyznaczanie trasy, 394
 - uruchamianie aplikacji systemowych, 385
 - URI mailto, 385, 386
 - URI sms, 388
 - URI tel, 387
 - uruchamianie YouTube, 388
 - uruchamianie APP Store, 389
 - uzyskiwanie URL, 389
 - wyświetlanie strony, 390
 - uruchamianie Safari, 316
 - wczytywanie dołączonego pliku HTML, 326
 - wczytywanie lokalnych stron WWW, 326
 - wielozadaniowość, 85
 - wsparcie dla wyświetlacza Retina, 242
 - wybór docelowych urządzeń, 238, 239
 - wykrywanie bieżącego urządzenia, 239
 - wykrywanie urządzenia z wyświetlaczem Retina, 242
 - wyłącznik aplikacji, 89, 117
 - zamykanie aplikacji, 87
 - zapisywanie plików, 377
 - ciągłość zapisu, 381
 - formaty danych, 380
 - monitorowanie postępu, 380
 - zapis synchroniczny, 380
 - zapisywanie stanu aplikacji, 115
 - zmiany orientacji ekranu, 255
 - blokowanie, 261
 - reakcja, 258
 - wykrywanie, 256
 - zrzut ekranu strony WWW, 334
 - App Development Overview, 31
 - App Store, 22
 - UDID Sender, 46
 - uruchamianie, 389
 - App Store Review Guidelines, 28
 - appendText(), 230
 - Array, 436, 438
 - attachCamera(), 302
 - autoCapitalize, 235
 - autoCorrect, 235
- B**
- beforeOrientation, 257
 - Bitmap.bitmapData, 199
 - bitmapa, 168, 170, 192
 - animowanie, 205
 - sekwencje animacji, 207
 - tworzenie animacji, 205

- buforowanie, 181
 - buforowanie pól tekstowych, 230
 - GPU-Blend, 186
 - GPU-Vector, 186
 - obiekty potomne, 185
 - potencjalne pułapki, 185
 - przekształcenia, 185
 - wzrost wydajności, 183
 - buforowanie z użyciem macierzy, 187
 - GPU-Blend, 191
 - korzyści, 189
 - właściwości 3D, 191
 - wybór macierzy, 190
 - wzrost wydajności, 188
 - dołączenie do aplikacji, 197
 - dostęp do danych, 199
 - ograniczenia wymiarów, 195
 - porównanie wydajności, 204
 - porównanie wymagań pamięci, 204
 - przechowywanie w GPU, 195
 - rozmiar obrazów, 195
 - tryb GPU, 195
 - usuwanie danych, 199
 - wczytanie, 197
 - wczytywanie na żądanie, 196, 198
 - wyświetlanie, 193, 197
 - zastępowanie pól tekstowych, 230
 - zestaw sprite'ów, 200
 - Stage 3D, 208
 - tworzenie, 204
 - BitmapData, 206, 290
 - BitmapData.copyPixels(), 204
 - BitmapData.dispose(), 199
 - BitmapData.draw(), 290
 - bitmapLoaded(), 198, 205
 - browseForImage(), 294
 - Bubble, 90
 - Button, 112
 - Button.as, 113
 - buttonPressed(), 114, 116
 - ByteArray, 304
 - bytesAvailable, 383
 - bytesLoaded, 384
 - bytesPending, 380
 - bytesTotal, 380, 384
- C**
- cacheAsBitmap, 183, 230
 - cacheAsBitmapMatrix, 188, 190, 230
 - Camera.getCamera(), 302, 303
 - CameraRoll, 290
 - CameraRoll.addBitmapData(), 290
 - CameraRoll.supportsAddBitmapData, 290
 - CameraRoll.supportsBrowseForImage, 294
 - CameraUI, 347
 - Capabilities.screenResolutionX, 240, 249
 - Capabilities.screenResolutionY, 240, 249
 - captured(), 297, 299
 - certyfikat dystrybucyjny, 38
 - certyfikat P12, 22, 38
 - system MAC OS, 41
 - system Microsoft Windows, 39
 - certyfikat programisty, 32
 - klucz prywatny, 33
 - klucz publiczny, 33
 - wniosek o podpisanie, 32
 - system MAC OS X, 34
 - system Microsoft Windows, 32
 - close(), 332, 353, 379
 - connect(), 340, 345
 - CPU, 171, 174, 175
 - komponowanie sceny, 174
 - podsumowanie, 179
 - rasteryzacja, 174
 - renderowanie wektorów, 180
 - createDirectory(), 376
 - createHtml(), 332
 - createMonkey(), 202, 203
 - czcionki iOS, 210
 - korzyści użycia, 211
 - krój czcionek, 212
 - krój pisma, 212
 - lista czcionek, 212, 216, 234
 - Fast Fonts, 212
 - Fonts, 212
 - osadzanie w aplikacji, 214, 215
 - trudności, 211
- D**
- dataCaptured(), 300
 - debugowanie, 104
 - etapy, 104
 - zdalny debugger, 105
 - próby połączenia, 107
 - deleteDirectory(), 376
 - deleteDirectoryAsync(), 376
 - deleteFile(), 333
 - deleteFileAsync(), 333, 384
 - describe(), 444

deviceOrientation, 258
 Dice, 448
 Dice.ROLLED, 448
 DisplayObject, 183, 188, 341
 cacheAsBitmap, 183
 cacheAsBitmapMatrix, 188
 dispose(), 203, 319
 documentsDirectory, 332
 dokument FLA, 60
 draw(), 291
 drawViewPortToBitmapData(), 335
 dystrybucyjny profil informacyjny, 53
 dźwięk w iOS, 338
 kontrola odtwarzania, 362, 363
 głośność, 365
 kontrola globalna, 366
 odtworzenie pliku od początku, 365
 odtworzenie pojedynczego dźwięku, 364
 panoramowanie, 365
 wznowienie odtwarzania, 365
 zatrzymanie odtwarzania, 365
 odtworzenie osadzonego dźwięku, 355, 356
 miejsce rozpoczęcia odtwarzania, 357
 odtworzenie w pętli, 357
 osadzanie w bibliotece, 353, 355
 pliki MP3, 358
 długość dźwięku, 361
 dołączanie pliku do aplikacji, 358
 metadane, 361
 monitorowanie wczytywania, 360
 odtworzenie dźwięku w tle, 361
 odtworzenie pliku, 358
 zwiększanie czasu bufora, 360

E

EncryptedLocalStore, 381
 ENTER_FRAME, 92, 95, 229
 enterFrame(), 92, 93
 ErrorEvent.ERROR, 291, 320, 326
 Event, 85, 88
 Event.CANCEL, 294
 Event.CLOSE, 379
 Event.COMPLETE, 198, 290, 295, 320, 360,
 376, 383
 Event.DEACTIVATE, 86, 87
 Event.ENTER_FRAME, 89, 92, 205
 Event.EXITING, 89
 Event.ID3, 361

Event.SOUND_COMPLETE, 365
 EXITING, 88
 exiting(), 88, 94

F

Fast Fonts, 212
 File, 331, 376
 File.deleteFile(), 384
 File.resolvePath(), 378
 fileLoaded(), 382
 FileMode.APPEND, 384
 FileMode.UPDATE, 384
 fileSaved(), 382
 FileStream, 331, 380
 FileSystem.close(), 383
 fl.motion.easing.Sine, 158
 fl.transitions.Tween, 158
 Flash Builder, 82
 Flash Professional, 22, 38, 58
 Adobe AIR, 58
 AIR Development Tool, 58
 AIR for iOS, 58, 61
 dokument FLA, 60
 stół montażowy, 62
 ustawienia środowiska, 67, 71
 AIR SDK, 58
 instalacja, 59
 nadpisywanie, 60
 aktualizacja, 59
 Biblioteka, 120
 Buforuj jako bitmapę, 182
 Buforuj jako mapę bitową, 64, 66, 182
 certyfikat P12, 38
 Częstotliwość próbkowania, 354
 debugowanie, 104
 etapy, 104
 zdalny debugger, 105
 dołączanie ikon, 99
 rodzaje ikon, 100
 Eksportuj dla ActionScript, 90
 Eksportuj jako bitmapę, 124
 Eksportuj PNG, 122
 gesty, 152
 machnięcie, 156
 obracanie obiektu, 162
 zmiana rozmiaru obiektu, 164
 przewijanie obiektu, 159
 włączenie obsługi, 154

- grafika rastrowa, 168
- grafika wektorowa, 168
- identyfikator aplikacji, 72
- Interpreter Mode, 75
- iPhone OS, 61
- kompilacja projektu FLA, 73
 - Interpreter Mode, 75
- konwersja na platformę iOS, 110
 - animacje na osi czasu, 124
 - dołączanie klasy do przycisku, 112
 - dostosowanie do rozmiarów ekranu, 124
 - dwustanowy przycisk, 113
 - przekształcenie grafiki wektorowej na rastrową, 121
 - przystosowanie do interfejsu dotykowego, 110
 - spłaszczenie listy wyświetlania, 119
 - tworzenie klipu filmowego przycisku, 111
 - wstępny rendering efektów graficznych, 125
- Konwertuj na bitmapę, 124
- maskowanie obiektów, 129
 - usunięcie warstwy maski, 130
- Multi-touch, 138
- narzędzia, 128, 210
 - Usuń punkt kontrolny (-), 128
 - Zaznaczenie cząstkowe (A), 128
 - Tekst (T), 210
- optymalizacja kształtów wektorowych, 128
- Osadzanie czcionek, 214, 215
- PFI, 58
- pętla uaktualnień, 92
- pliki SWC, 132
 - publikowanie, 133
 - tworzenie, 133
- pliki SWF, 132
- pobranie danych wideo z aparatu, 301
- Profil udostępniania, 71
- przekształcenie w bitmapę, 121
- publikowanie aplikacji iOS, 40
 - certyfikat P12, 40
- renderowanie grafiki, 168
 - Automatycznie, 176
 - buforowanie bitmap, 181, 183, 185
 - buforowanie z użyciem macierzy, 187, 188, 189
 - GPU-Vector, 180
 - komponowanie sceny, 174
 - obniżenie jakości, 170
 - rasteryzacja, 174
 - renderowanie w GPU, 172
 - Stage 3D, 181
 - test wydajności, 168, 169
 - tryb GPU-Blend, 174
 - tryb GPU-Vector, 176, 179
 - tryby renderowania, 173, 179
- skalowanie bitmap, 126
- spłaszczenie listy wyświetlania, 119
 - rozgrupowanie instancji, 120
- Text Layout Framework, 213
- Ustawienia publikowania, 133, 216
- Ustawienia środowiska AIR for iOS, 67, 95, 99
 - Dołączone pliki, 70
 - Nazwa aplikacji, 68
 - Numer wersji, 68
 - Orientacja, 68
 - Pełny ekran, 69
 - Renderowanie, 68, 177
 - Urządzenie, 68
- Ustawienia telefonu iPhone, 172, 173
- Utwórz nowy symbol, 111
- Użyj czcionek urządzenia, 211
- Wdrożenie, 71
 - typy wdrożenia, 72
- Właściwości dźwięku, 354
- Właściwości symbolu, 90
- Zapamiętaj hasło dla tej sesji, 71
- zmiana rozmiarów bitmap, 126
- flash.filesystem.File, 378
- flash.display.Bitmap, 199, 204, 336
- flash.display.BitmapData, 194, 199, 204, 336
- flash.display.DisplayObject, 341
- flash.display.Loader, 199, 203
- flash.display.MovieClip, 92
- flash.display.Sprite, 92, 147, 149
- flash.display.StageAlign, 240, 244, 249, 255
- flash.display.StageOrientation, 257, 261
- flash.display.StageQuality, 128
- flash.display.StageScaleMode, 240, 244, 249, 254
- flash.events.ErrorEvent.ERROR, 298
- flash.events.FocusEvent, 223
- flash.events.GeolocationEvent, 267, 272
- flash.events.GesturePhase, 161
- flash.events.LocationChangeEvent, 324
- flash.events.MouseEvent, 141, 155
- flash.events.NetStatusEvent, 353
- flash.events.SampleDataEvent, 308
- flash.events.SoftKeyboardEvent, 227

- flash.events.StageVideoEvent, 347
- flash.events.StatusEvent, 277
- flash.events.TouchEvent, 144
- flash.events.TransformGestureEvent, 160
- flash.filesystem.File, 327, 332, 333, 375, 376, 379
- flash.filesystem.FileMode, 379
- flash.filesystem.FileStream, 333, 378, 379
- flash.geom.Matrix, 189, 190
- flash.media.Camera, 301, 302
- flash.media.CameraRoll, 291, 294, 295
- flash.media.CameraUI, 298
- flash.media.MediaPromise, 295, 298
- flash.media.MediaType, 298
- flash.media.Microphone, 307, 308
- flash.media.Sound, 312, 356, 359, 365
- flash.media.SoundChannel, 312, 365
- flash.media.SoundLoaderContext, 360
- flash.media.SoundMixer, 366
- flash.media.SoundTransform, 365
- flash.media.StageVideo, 346
- flash.media.StageWebView, 319, 324, 327
- flash.media.Video, 302, 341
- flash.net.navigateToURL, 317
- flash.net.NetConnection, 341, 346
- flash.net.NetStream, 341, 346
- flash.net.registerClassAlias(), 117
- flash.net.URLRequest, 317
- flash.sampler.getSize(), 445, 446
- flash.sensors.Accelerometer, 264
- flash.sensors.Geolocation, 264, 267, 272
- flash.system.Capabilities, 240, 244, 249
- flash.text.Font, 213
- flash.text.Font.enumerateFonts(), 234
- flash.text.ReturnKeyLabel, 234
- flash.text.SoftKeyboardType, 234
- flash.text.TextField, 223
- flash.ui.Multitouch, 139, 154
- flash.ui.MultitouchInputMode, 139, 154
- flash.utils.ByteArray, 308, 333
- flash.utils.getTimer(), 436, 437
- flush(), 118
- FOCUS_IN, 223
- FOCUS_OUT, 222
- FocusEvent, 222
- focusLost(), 222
- Font.enumerateFonts(), 213
- fontFamily, 233
- Fonts, 212
- funkcje
 - accUpdated(), 279, 285
 - activate(), 225
 - bitmapLoaded(), 198
 - buttonPressed(), 114, 116
 - enterFrame(), 92
 - fileLoaded(), 382
 - fileSaved(), 382
 - flash.net.registerClassAlias(), 117
 - flash.sampler.getSize(), 445
 - flash.utils.getTimer(), 436
 - focusLost(), 222
 - geoUpdated(), 271, 272
 - getLocal(), 117
 - getSize(), 445
 - initSharedObject(), 117
 - navigateToURL(), 316, 317, 386, 388, 391, 392
 - orientationChanged(), 257
 - orientationChanging(), 257
 - pan(), 160
 - photoLoaded(), 295
 - photoSelected(), 293
 - playSampleData(), 311
 - pressed(), 297
 - SampleData(), 307
 - startTouchDrag(), 147
 - statusUpdated(), 276
 - stopTouchDrag(), 147
 - swipe(), 158
 - touchBegin(), 147
 - touchMove(), 152
 - update(), 229
 - zoom(), 165

G

- geokodowanie, 268
- Geolocation, 266, 394
- Geolocation.isSupported, 267
- Geolocation.Muted, 277
- Geolocation.setRequestedUpdateInterval(), 266
- GeolocationEvent, 271
- GeolocationEvent.altitude, 267
- GeolocationEvent.heading, 271
- GeolocationEvent.speed, 271
- GeolocationEvent.UPDATE, 266, 276
- GeolocationEvent.verticalAccuracy, 267
- geoUpdated(), 266, 271, 272

GESTURE_PAN, 160
 GesturePhase.BEGIN, 161
 GesturePhase.END, 161
 GesturePhase.UPDATE, 161
 GESTURE_SWIPE, 158
 GestureEvent.GESTURE_TWO_FINGER_TAP, 155
 gesty, 152
 iOS Human Interface Guidelines, 154
 machnięcie, 156
 w pionie, 158
 obracanie obiektu, 162
 zmiana rozmiaru obiektu, 164
 przewijanie obiektu, 159
 fazy gestu, 161
 przewijanie jednym palcem, 162
 włączenie obsługi, 154
 Multitouch.inputMode, 154
 zdarzenia myszy, 155
 zdarzenia związane z gestami, 155
 getCamera(), 301
 getLocal(), 117
 getMilesPerHour(), 272
 getSize(), 445
 getTimer(), 437
 GPU-Blend, 171, 174, 175, 179
 buforowanie bitmap, 186
 buforowanie z użyciem macierzy, 192
 komponowanie sceny, 174
 podsumowanie, 179
 ponowna rasteryzacja, 175
 przechowywanie bitmap, 195
 rasteryzacja, 174
 renderowanie wektorów, 180
 GPU-Vector, 176, 179
 buforowanie bitmap, 186
 komponowanie sceny, 179
 podsumowanie, 179
 przechowywanie bitmap, 195
 rasteryzacja, 179
 renderowanie wektorów, 180
 grafika rastrowa, *Patrz* bitmapa
 grafika wektorowa, 168, 170

H

historyBack(), 323
 historyForward(), 323

I

identyfikator aplikacji, 47, 48, 72
 edycja, 50
 identyfikator pakunku, 48
 dystrybucja aplikacji, 49
 konwencje nazewnictwa, 49
 określony identyfikator, 47
 wieloznaczny identyfikator, 47
 tworzenie, 47
 <infoAdditions>, 102, 374
 Information Property List, 103
 initWithSharedObject(), 117
 InteractiveObject, 140, 144, 157
 Interpreter Mode, 405, 421
 kompilacja aplikacji, 423
 włączanie, 422
 IOErrorEvent.IO_ERROR, 199, 360, 379
 iOS App Workflow Guide, 31, 37
 iOS Dev Center, 27, 28
 iOS Human Interface Guidelines, 139
 iOS Developer Program, 22
 Enroll Now, 23
 iOS Dev Center, 27, 28
 App Store Review Guidelines, 28
 iOS Human Interface Guidelines, 28
 iOS Provisioning Portal, 27, 29
 Apple Developer Technical Support, 31
 Member Center, 31
 Resources, 30
 praca zespołowa, 28
 rejestracja, 23
 rejestracja jako programista Apple, 24
 iOS Human Interface Guidelines, 28, 101, 139, 154
 iOS Provisioning Portal, 27, 29
 administrator zespołu, 38
 agent zespołu, 38
 Apple Developer Technical Support, 31
 certyfikat programisty, 32, 35
 Certificates, 35
 Development, 35
 Expiration Date, 37
 Request Certificate, 35
 członek zespołu, 38
 Member Center, 31
 programistyczny profil informacyjny, 50
 Resources, 30
 App Development Overview, 31
 iOS App Workflow Guide, 31, 37

<iPhone>, 102, 374
 iPhone Configuration Utility, 78
 isBuffering, 360
 isHistoryBackEnabled, 323
 isHistoryForwardEnabled, 323
 isPrimaryTouchPoint, 145
 iTunes, 75

K

KeyboardEvent.KEY_DOWN, 235
 KeyboardEvent.KEY_UP, 235
 klasy, 80, 81

- Accelerometer, 279
- air.net.SocketMonitor, 371
- Array, 436, 438
- Bubble, 90
- Button, 112
- Button.as, 113
- ByteArray, 304
- CameraRoll, 290
- CameraUI, 347
- Dice, 448
- DisplayObject, 183, 188, 341
- EncryptedLocalStore, 381
- Event, 85, 88
- File, 331, 376
- FileStream, 331, 380
- fl.motion.easing.Sine, 158
- fl.transitions.Tween, 158
- flash.display.BitmapData, 194
- flash.display.DisplayObject, 341
- flash.display.Sprite, 147, 149
- flash.display.StageAlign, 255
- flash.display.StageOrientation, 257, 261
- flash.display.StageQuality, 128
- flash.display.StageScaleMode, 254
- flash.events.GesturePhase, 161
- flash.events.MouseEvent, 141, 155
- flash.events.TouchEvent, 144
- flash.filesystem.File, 327, 332, 375, 378
- flash.filesystem.FileStream, 378
- flash.geom.Matrix, 189
- flash.media.Camera, 301
- flash.media.CameraRoll, 294
- flash.media.CameraUI, 298
- flash.media.MediaType, 298
- flash.media.Microphone, 307
- flash.media.Sound, 359
- flash.media.SoundMixer, 366
- flash.media.Video, 302
- flash.sensors.Accelerometer, 264
- flash.sensors.Geolocation, 264
- flash.text.ReturnKeyLabel, 234
- flash.text.SoftKeyboardType, 234
- Font, 213
- Geolocation, 266, 394
- konstruktor klasy, 81
- Loader, 198
- Main, 114, 144
- Main.as, 115, 171
- MediaPromise, 295
- Microphone, 304
- MonkeyBitmapData, 194
- MovieClip, 92
- NativeApplication, 84, 86
- nazwa klasy, 81
- NetConnection, 303, 340, 345
- NetStream, 303, 340, 345, 352
- pisanie własnej klasy, 90
- ProgressEvent, 361
- Rectangle, 445
- SharedObject, 115, 116, 117, 381
- SoftKeyboardEvent, 227
- Sound, 311, 357
- SoundChannel, 311
- SoundEffect, 355
- Sprite, 92
- StageText, 231, 233
- StageVideo, 345
- StageWebView, 318, 319
- TextField, 230
- tworzenie, 80
- URLMonitor, 368, 371
- Vector, 438, 439, 440
- Video, 340

 klawiatura wirtualna, 217

- needsSoftKeyboard, 220
- przewijanie podczas aktywacji, 223
- rodzaje, 234
 - wybór rodzaju, 234
- typy, 220
- uruchamianie automatyczne, 220
- uruchamianie programistyczne, 220
- wejściowe pole tekstowe, 218
- włączenia automatycznego przewijania, 224

 kompilacja ahead-of-time, 75
 kompilator LLVM, 75

kompilator Mac OS X, 406
 dołączanie innych plików źródłowych, 409
 opcje debugowania, 410
 opis bloków polecenia, 407
 tworzenie skryptu powłoki, 410
 typy wdrożenia aplikacji, 408
 kompilator Microsoft Windows, 400
 dołączanie innych plików źródłowych, 404
 opcje debugowania, 404
 opis bloków polecenia, 401
 tworzenie pliku wsadowego, 405
 typy wdrożenia aplikacji, 402

L

lash.display.Bitmap, 194
 launch(), 298, 299
 load(), 198
 Loader, 198
 Loader.load(), 295
 Loader.loadFilePromise(), 295
 loadFilePromise(), 295, 299
 loadHtml(), 333
 loadPCMFromByteArray(), 361
 loadString(), 333
 loadURL(), 319, 323
 localX, 145
 localY, 145
 LOCATION_CHANGING, 324
 locationChanged(), 323
 LocationChangeEvent, 324
 LocationChangeEvent.LOCATION_CHANGE,
 324

M

Main, 81, 114, 144
 Main.as, 115, 171
 MediaEvent, 294
 MediaEvent.COMPLETE, 298
 MediaEvent.SELECT, 294
 MediaPromise, 295
 metody
 Accelerometer.setRequestedUpdateInterval(),
 279
 activate(), 86
 addBitmapData(), 290
 addChild(), 302
 addEventListener(), 452

appendText(), 230
 attachCamera(), 302
 BitmapData.copyPixels(), 204
 BitmapData.dispose(), 199
 BitmapData.draw(), 290
 bitmapLoaded(), 205
 browseForImage(), 294
 Camera.getCamera(), 302, 303
 CameraRoll.addBitmapData(), 290
 captured(), 297
 close(), 332, 353, 379
 connect(), 340, 345
 createDirectory(), 376
 createHtml(), 332
 createMonkey(), 202, 203
 dataCaptured(), 300
 deleteDirectory(), 376
 deleteDirectoryAsync(), 376
 deleteFile(), 333
 deleteFileAsync(), 333
 describe(), 444
 dispose(), 203, 319
 draw(), 291
 drawViewPortToBitmapData(), 335
 enterFrame(), 93
 exiting(), 88, 94
 File.resolvePath(), 378
 FileSystem.close(), 383
 flush(), 118
 Font.enumerateFonts(), 213
 geoUpdated(), 266
 getCamera(), 301
 getLocal(), 117
 getMilesPerHour(), 272
 getTimer(), 437
 historyBack(), 323
 historyForward(), 323
 initSharedObject(), 117
 launch(), 298, 299
 load(), 198
 Loader.load(), 295
 Loader.loadFilePromise(), 295
 loadFilePromise(), 295, 299
 loadHtml(), 333
 loadPCMFromByteArray(), 361
 loadString(), 333
 loadURL(), 319, 323
 locationChanged(), 323
 Microphone.getMicrophone(), 307

- metody
 - NetStream.play(), 341, 347
 - open(), 299, 332, 380, 384
 - openAsync(), 379, 383
 - pause(), 352
 - play(), 311, 346, 357, 360, 365
 - preventDefault(), 261, 324
 - readFloat(), 312
 - readUTFBytes(), 383
 - reload(), 324
 - removeEventListener(), 452
 - requestSoftKeyboard(), 220
 - resolvePath(), 327, 376
 - resume(), 352
 - roll(), 447, 448
 - savePreferences(), 380
 - saveSharedObject(), 117
 - seek(), 352
 - setMode(), 302
 - setRequestedUpdateInterval(), 285
 - setSilenceLevel(), 308
 - setUpControls(), 350
 - sliceSpriteSheet(), 202, 203
 - Sound.play(), 364
 - StageWebView.loadURL(), 326
 - start(), 371
 - startRecording(), 307, 308
 - startTouchDrag(), 147, 148
 - stop(), 324
 - stopPropagation(), 453
 - stopRecording(), 308
 - stopTouchDrag(), 147
 - System.pauseForGCIfCollectionImminent(), 442
 - touchEnd(), 144
 - update(), 93, 95, 142
 - URLMonitor.stop(), 371
 - writeBytes(), 333
 - writeFile(), 332
 - writeFloat(), 312
 - writePng(), 332
 - writeUTFBytes(), 332, 333, 379
 - Microphone, 304
 - Microphone.getMicrophone(), 307
 - mikrofon, 287
 - częstotliwość próbkowania dźwięku, 307
 - konwersja częstotliwości próbkowania, 313
 - nagrywanie dźwięku, 304
 - natężenie dźwięku, 308
 - odtwarzanie nagranych dźwięku, 309
 - udostępnianie strumienia dźwięku, 309
 - współczynnik wzmocnienia, 307
 - MonkeyBitmapData, 194
 - MOUSE_UP, 115, 450
 - MouseEvent.MOUSE_UP, 114
 - MovieClip, 92, 140, 144, 157
 - Multi-touch, 137, 141
 - globalne współrzędne dotknięcia, 145
 - identyfikator punktu kontaktu, 145
 - iOS Human Interface Guidelines, 139
 - lokalne współrzędne dotknięcia, 145
 - Multitouch.inputMode, 139, 154
 - Multitouch.maxTouchPoints, 140
 - Multitouch.supportsTouchEvents, 139
 - MultitouchInputMode.NONE, 141
 - MultitouchInputMode.TOUCH_POINT, 139, 154
 - obsługa wielu zdarzeń dotyku, 143
 - pierwszy punkt kontaktu, 145
 - przeciąganie wyświetlonych obiektów, 146
 - punkt kontaktu, 140
 - cel dotknięcia, 140
 - śledzenie ruchu, 150
 - włączenie trybu wejściowego, 138
 - flash.ui.Multitouch, 139
 - flash.ui.MultitouchInputMode, 139
 - zdarzenia dotyku, 140
 - zdarzenie myszy, 141
 - zsuniecie palca, 144
 - Multitouch.supportedGestures, 155
 - Multitouch.supportsGestureEvents, 154
 - MultitouchInputMode.GESTURE, 155
 - MVC, 249
 - PureMVC, 250
 - Robotlegs, 250
- ## N
- NativeApplication, 84, 86
 - Event.DEACTIVATE, 86, 87
 - EXITING, 88
 - systemIdleMode, 84
 - SystemIdleMode.KEEP_AWAKE, 84
 - NativeApplication.systemIdleMode, 342
 - nativePath, 376
 - navigateToURL(), 316, 317, 386, 388, 391, 392
 - needsSoftKeyboard, 220

NET_STATUS, 351
 NetConnection, 303, 340, 345
 NetStatusEvent.NET_STATUS, 342, 347
 NetStream, 303, 340, 345, 352
 NetStream.play(), 341, 347

O

OAuth, 318, 321
 odbiornik GPS, 263, 264

- bieżąca wysokość, 267
- dokładność danych lokalizacyjnych, 267
- prędkość i kierunek poruszania się, 270

 onCuePoint, 346
 onMetaData, 341, 346
 onXMPPData, 341, 346
 open(), 299, 332, 380, 384
 openAsync(), 379, 383
 OutputProgressEvent.OUTPUT_PROGRESS, 380

P

Packager for iPhone, *Patrz* PFI
 pan(), 160
 pause(), 352
 PFI, 58, 424
 photoLoaded(), 293, 295
 photoSelected(), 293, 294
 play(), 311, 346, 357, 360, 365
 playSampleData(), 311
 plik deskryptora aplikacji, 101, 372

- deklarowanie rozszerzenia, 417
- edycja, 102
- Information Property List, 103
- określenie wymagań sprzętowych, 395
- otwieranie, 102
- UIStatusBarStyle, 103
- ustawianie stałego połączenia Wi-Fi, 372, 374

 pliki SWC, 132

- publikowanie, 133
- dołączanie, 133, 368, 412
- tworzenie, 133

 pliki SWF, 132, 136

- osadzanie czcionek, 215

 pollInterval, 372
 pressed(), 289, 297, 298
 preventDefault(), 261, 324
 profil informacyjny Ad Hoc, 53

programistyczny profil informacyjny, 22, 50

- edycja, 52
- rejestracja na urządzeniu, 53
- tworzenie, 51
- usuwanie z urządzenia, 54

 ProgressEvent, 361
 ProgressEvent.PROGRESS, 360, 384

R

readFloat(), 312
 readUTFBytes(), 383
 Rectangle, 445
 reload(), 324
 removeEventListener(), 452
 renderowanie, 168

- Automatycznie, 176
- buforowanie bitmap, 181, 183, 185
- buforowanie z użyciem macierzy, 187, 188, 189
- GPU-Vector, 180
- komponowanie sceny, 174
- obniżenie jakości, 170
- rasteryzacja, 174
- renderowanie plików H.264, 346
- renderowanie plików FLV, 341
- renderowanie programistyczne, 171
- renderowanie strony WWW, 318
 - dołączanie banerów reklamowych, 320
- renderowanie w GPU, 172
- Stage 3D, 181
- test wydajności, 168, 169
- tryby renderowania, 173, 179
 - GPU-Blend, 174
 - GPU-Vector, 176, 179

 requestSoftKeyboard(), 220
 resolvePath(), 327, 376
 resume(), 352
 returnKeyLabel, 234
 roll(), 447, 448

S

Safari, 23, 316
 SAMPLE_DATA, 307, 311
 SampleData(), 307
 SampleDataEvent, 312
 SampleDataEvent.SAMPLE_DATA, 307

- saved(), 289
 - savePreferences(), 380
 - saveSharedObject(), 117
 - seek(), 352
 - setMode(), 302
 - setRequestedUpdateInterval(), 285
 - setSilenceLevel(), 308
 - setupControls(), 350
 - SharedObject, 115, 116, 117, 381
 - singleton, 84
 - sliceSpriteSheet(), 202, 203
 - SOFT_KEYBOARD_ACTIVATE, 225, 226
 - SOFT_KEYBOARD_DEACTIVATE, 226
 - softKeyboardBehavior, 225
 - SoftKeyboardEvent, 227
 - SoftKeyboardEvent.SOFT_KEYBOARD_ACTIVATE, 235
 - SoftKeyboardEvent.SOFT_KEYBOARD_ACTIVATING, 235
 - SoftKeyboardEvent.SOFT_KEYBOARD_DEACTIVATE, 235
 - softKeyboardType, 234
 - Sound, 311, 357
 - Sound.play(), 364
 - SOUND_COMPLETE, 311
 - SoundChannel, 311, 357, 364
 - SoundEffect, 355
 - Sprite, 92, 140, 144, 157
 - Stage, 140, 144, 157
 - Stage 3D, 181, 208
 - zestaw sprite'ów, 208
 - Stage.align, 255
 - Stage.autoOrients, 254
 - Stage.deviceOrientation, 261
 - Stage.quality, 128, 170
 - Stage.scaleMode, 254
 - Stage.setOrientation(), 261
 - Stage.softKeyboardRect, 226, 227
 - StageText, 231, 233
 - StageVideo, 345
 - StageVideoEvent.RENDER_STATE, 347
 - StageWebView, 318, 319
 - StageWebView.isSupported, 320
 - StageWebView.loadURL(), 326
 - StageWebView.viewPort, 335
 - stageX, 145
 - stageY, 145
 - start(), 371
 - startRecording(), 307, 308
 - startTouchDrag(), 147, 148
 - bounds, 149
 - lockCenter, 149
 - touchPointID, 149
 - STATUS, 276
 - StatusEvent.STATUS, 276, 371
 - statusUpdated(), 276
 - stop(), 324
 - stopPropagation(), 453
 - stopRecording(), 308
 - stopTouchDrag(), 147
 - stół montażowy, 62
 - budowa prostej sceny, 64
 - jakość renderowania, 171
 - kolor stołu, 261
 - tryby skalowania, 249, 254
 - wymiary, 63
 - wyrównanie zawartości stołu, 255
 - zmiana orientacji stołu, 261
 - supportsOrientationChange, 254
 - swipe(), 158
 - System.pauseForGCIfCollectionImminent(), 442
 - systemIdleMode, 84
- ## T
- TestFlight, 78
 - Text Layout Framework, 213
 - TextField, 230
 - TOUCH_BEGIN, 144
 - TOUCH_END, 144
 - TOUCH_MOVE, 150
 - touchBegin(), 147, 148
 - touchEnd(), 144, 148
 - TouchEvent.TOUCH_MOVE, 150, 151
 - TouchEvent.TOUCH_ROLL_OUT, 144
 - touchMove(), 152
 - touchPointID, 145, 148
 - trace(), 104
 - TransformGestureEvent, 158
 - phase, 161
 - TransformGestureEvent.GESTURE_PAN, 155, 159, 160
 - TransformGestureEvent.GESTURE_ROTATE, 155, 162, 163
 - TransformGestureEvent.GESTURE_SWIPE, 155, 156, 157
 - TransformGestureEvent.GESTURE_ZOOM, 155, 164, 165

U

UDID, 43
 UDID Sender, 46
 UIRequiredDeviceCapabilities, 396
 UIRequiresPersistentWiFi, 374
 UIStatusBarStyle, 103
 UPDATE, 266, 271, 279
 update(), 93, 95, 142, 229
 URI mailto, 385, 386
 dostępne parametry, 387
 URI sms, 388
 URI tel, 387
 URLMonitor, 368, 371
 URLMonitor.running, 371
 URLMonitor.stop(), 371
 URLRequest, 360
 Ustawienia środowiska AIR for iOS, 71

V

Vector, 438, 439, 440
 Video, 340
 videoHeight, 347
 videoWidth, 347

W

węzły, 102
 <infoAdditions>, 102
 <iPhone>, 102
 wideo w iOS, 338
 kontrola odtwarzania, 348, 352
 ponowne odtwarzanie pliku, 352
 wstrzymanie odtwarzania pliku, 353
 wznowienie odtwarzania pliku, 352
 pliki FLV, 338
 blokowanie ekranu, 342
 dołączanie do aplikacji, 338
 dołączanie wielu plików, 342
 maksymalizacja wydajności odtwarzania,
 342
 metadane, 341
 odtwarzanie pliku dostępnego on-line, 341
 odtwarzanie pliku lokalnego, 339
 punkty sygnalizacji, 341
 renderowanie w CPU, 341

pliki H.264, 343
 dołączanie do aplikacji, 343
 dołączanie wielu plików, 348
 kodowanie, 348
 metadane, 346
 odtwarzanie pliku dostępnego on-line, 347
 odtwarzanie pliku lokalnego, 344
 punkty sygnalizacji, 346
 renderowanie w GPU, 346
 wyznaczanie rozmiarów klipu, 347

writeBytes(), 333
 writeFile(), 332
 writeFloat(), 312
 writePng(), 332
 writeUTFBytes(), 332, 333, 379
 wyłącznik aplikacji, 89, 117
 wyświetlacz Retina, 241
 ikona ekranu głównego, 245
 obraz rozruchowy aplikacji, 245
 rozdzielczość, 243
 wsparcie dla rozdzielczości, 242
 wykrywanie, 242
 wyświetlanie stron WWW, 316
 adres URL bieżącej strony, 324
 dołączanie banerów reklamowych, 320
 dynamiczne generowanie strony WWW, 328
 ponowne wczytanie strony, 324
 przeglądanie historii stron, 321
 renderowanie lokalnych stron WWW, 326
 renderowanie strony WWW, 318
 Safari, 316
 uruchamianie, 316
 wsparcie usługi OAuth, 321
 zatrzymywanie wczytywania strony, 324
 zmiana adresu URL strony, 324
 zrzut ekranu strony WWW, 334
 zwalnianie pamięci, 333
 wzorzec projektowy Model-Widok-Kontroler,
Patrz MVC

Z

zdarzenia
 AccelerometerEvent.UPDATE, 279, 281, 285
 captured(), 299
 Dice.ROLLED, 448
 ENTER_FRAME, 92, 95, 229
 ErrorEvent.ERROR, 291, 320

zdarzenia

- Event.CANCEL, 294
- Event.CLOSE, 379
- Event.COMPLETE, 198, 290, 295, 320, 360, 376, 383
- Event.DEACTIVATE, 86, 87
- Event.ENTER_FRAME, 89, 92, 205
- Event.EXITING, 89
- Event.ID3, 361
- Event.SOUND_COMPLETE, 365
- EXITING, 88
- FOCUS_IN, 223
- FOCUS_OUT, 222
- FocusEvent, 222
- GeolocationEvent.UPDATE, 266, 276
- GESTURE_PAN, 160
- GESTURE_SWIPE, 158
- GestureEvent.GESTURE_TWO_FINGER_TAP, 155
- IOErrorEvent.IO_ERROR, 199, 360, 379
- KeyboardEvent.KEY_DOWN, 235
- KeyboardEvent.KEY_UP, 235
- LOCATION_CHANGING, 324
- LocationChangeEvent.LOCATION_CHANGE, 324
- MediaEvent.COMPLETE, 298
- MediaEvent.SELECT, 294
- MOUSE_UP, 115, 450
- MouseEvent.MOUSE_UP, 114
- NET_STATUS, 351
- NetStatusEvent.NET_STATUS, 342, 347
- onCuePoint, 346
- onMetaData, 341, 346
- onXMPData, 341, 346
- ORIENTATION_CHANGE, 257
- ORIENTATION_CHANGING, 257
- OutputProgressEvent.OUTPUT_PROGRESS, 380
- photoLoaded(), 293
- photoSelected(), 294
- pressed(), 289, 298
- ProgressEvent.PROGRESS, 360, 384
- SAMPLE_DATA, 307, 311
- SampleDataEvent.SAMPLE_DATA, 307
- saved(), 289
- SOFT_KEYBOARD_ACTIVATE, 225, 226
- SOFT_KEYBOARD_DEACTIVATE, 226
- SoftKeyboardEvent.SOFT_KEYBOARD_ACTIVATE, 235
- SoftKeyboardEvent.SOFT_KEYBOARD_ACTIVATING, 235
- SoftKeyboardEvent.SOFT_KEYBOARD_DEACTIVATE, 235
- SOUND_COMPLETE, 311
- StageVideoEvent.RENDER_STATE, 347
- STATUS, 276
- StatusEvent.STATUS, 276, 371
- TOUCH_BEGIN, 144
- TOUCH_END, 144
- TOUCH_MOVE, 150
- touchBegin(), 148
- touchEnd(), 148
- TouchEvent.TOUCH_MOVE, 150, 151
- TouchEvent.TOUCH_ROLL_OUT, 144
- TransformGestureEvent.GESTURE_PAN, 155, 159, 160
- TransformGestureEvent.GESTURE_ROTATE, 155, 162, 163
- TransformGestureEvent.GESTURE_SWIPE, 155, 156, 157
- TransformGestureEvent.GESTURE_ZOOM, 155, 164, 165
- UPDATE, 266, 271, 279
- zoom(), 165

ż

żyroskop, 269

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



- 1. ZAREJESTRUJ SIĘ**
- 2. PREZENTUJ KSIĄŻKI**
- 3. ZBIERAJ PROWIZJĘ**

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

Tworzenie aplikacji dla iOS we Flashu

Receptury

Środowisko Adobe Flash cieszy się ogromną popularnością. Dzięki niemu użytkownicy mogli po raz pierwszy zobaczyć atrakcyjne wizualnie aplikacje internetowe. Firma Adobe poszła więc o krok dalej – przygotowała środowisko uruchomieniowe Adobe Air. Dzięki niemu raz napisaną aplikację można uruchamiać na różnych platformach. Teraz do tych platform dołączył iOS. Dla Ciebie oznacza to, że wystarczy opanowanie kilku sprawdzonych receptur, by zacząć błyskawicznie przygotowywać aplikacje na urządzenia mobilne spod szyldu Apple.

W książce „Tworzenie aplikacji dla iOS we Flashu. Receptury” bez trudu znajdziesz rozwiązania większości typowych problemów. W trakcie lektury nauczysz się wykrywać położenie urządzenia, odtwarzać pliki multimedialne oraz obsługiwać gesty i technologię multi-touch. Ponadto dowiesz się, jak przekształcić dotychczasowe projekty na takie, które uda się uruchomić na platformie iOS, oraz dostosować rozdzielczość w zależności od urządzenia. Książka ta jest idealną pozycją dla osób chcących przygotować uniwersalną aplikację na różne platformy.

Twórz uniwersalne aplikacje dzięki mobilnemu środowisku Adobe Air!



Sprawdź, jak w Adobe Air dla iOS:

- korzystać z geolokalizacji
- sterować za pomocą gestów i wykorzystać możliwości multi-touch
- odtwarzać pliki multimedialne
- wyświetlać strony WWW

helion.pl
księgarnia
internetowa

Nr katalogowy: 11713



Księgarnia internetowa
<http://helion.pl>



Zamówienia telefoniczne:

0 801 339900



0 601 339900



Helion

Sprawdź najnowsze promocje:

📍 <http://helion.pl/promocje>

📖 Książki najchętniej czytane:

📍 <http://helion.pl/bestsellery>

📄 Zamów informacje o nowościach:

📍 <http://helion.pl/nowosci>

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel.: 32 230 98 63

e-mail: helion@helion.pl

<http://helion.pl>

sięgnij po **WIĘCEJ**



KOD KORZYŚCI

ISBN 578-83-246-4993-8



Cena: 79,00 zł

Informatyka w najlepszym wydaniu