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

Helion

Christopher Caleb

[PACKT PUBLISHING]

Tytuł oryginału: Flash iOS Apps Cookbook

Tłumaczenie: Joanna Zatorska

ISBN: 978-83-246-4993-8

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.

Spis treści

Zespół

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

9

ROZDZIAŁ

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`.

Dokonaliśmy 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 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 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. Zacznij chodzić, a najlepiej pobiegnij. 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 ciągłe 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`.

Napiszemy 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` wyzwoli 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 urządzenia 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.addEventListener(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 nasłuchiwania 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 nagła 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 nagłe 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 potrząsania

Akcelerometr często służy do wykrywania potrząsania, 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 przesuwał klip w odpowiedzi na ruch urządzenia względem trzech osi. Gdy urządzenie zarejestruje ruch, aplikacja przejdzie do drugiej klatki klipu, aby poinformować użytkownika o wykryciu potrząsania.

Stół montażowy w tym projekcie jest zorientowany poziomo.

Jak to zrobić...

Wykonaj poniższe czynności, aby wykryć potrząsanie:

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` wyzwalanego 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 potrząsania.

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

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Z

Ż

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.

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

Twórz uniwersalne aplikacje dzięki mobilnemu środowisku Adobe Air!

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

Informatyka w najlepszym wydaniu

sięgnij po WIĘCEJ
KOD KORZYŚCI

ISBN 978-83-246-4993-8

Cena: 79,00 zł