8-Sensors_and_Multimedia

Core Motion

Core Motion fornisce un framework unificato per ricevere e processare dati provenienti dal dispositivo hardware, permette di accedere ai dati, raw e processati, utilizzando un interfaccia basata sui blocchi, di:

- Accelerometro, `CMAccelerometerData`
- Giroscopio, `CMGyroData`
- Magnetometro, `CMMagnetometerData`

CMMotionManager

Un oggetto di tipo `CMMotionManager` è la porta per i servizi di movimento forniti da iOS, quali dati su accelerometro, rotazione, magnetometro e molti altri, come ad esempio l'altitudine.

> Nota
> Questo tipo di oggetto è creato mediante `alloc/init`

> Nota
> All'interno dell'applicazione deve essere presente un solo `CMMotrionManager`

Dopo che è stato creato un oggetto di tipo `CMMotionManager` può essere configurato per impostare alcuni parametri come l'intervallo di aggiornamento per ogni tipo di movimento.
Dopo la sua creazione e configurazione è possibile chiedergli di mantenere di 4 tipi di moti:

- dati raw dell'accelerometro
- dati raw del giroscopio
- dati raw del magnetometro
- dati processati dei movimenti del dispositivo

Per ricevere i dati di movimento a intervalli specifici, l'app chiama un metodo "start" che accetta una coda di operazioni e un gestore di blocchi di un tipo specifico per l'elaborazione di tali aggiornamenti,. I dati passati sono:

```
startAccelerometerUpdatesToQueue:(NSOperationQueue *)
                                        withHandler:^(CMAccelerometerData *accelerometerData,
NSError *error)

startGyroUpdatesToQueue:(NSOperationQueue *)
                         withHandler:^(CMGyroData *gyroData, NSError *error)

startMagnetometerUpdatesToQueue:(NSOperationQueue *)
```

```
                                        withHandler:^(CMMagnetometerData *magnetometerData,
NSError *error)
```

Per terminare la ricezione degli aggiornamenti sono presenti dei metodi appropriati:
- `stopAccelerometerUpdates`
- `stopGyroUpdates`
- `stopMagnetometerUpdates`
- `stopDeviceMotionUpdates`

Core Motion Cookbook

È possibile impostare uno specifico intervallo di aggiornamento

Strumento	Proprietà	Metodo	Handler
Accelerometro	`accelerometerUpdateInterval`	`startAccelerometerUpdatesToQueue:withHandler:`	`CMAccelerometerHand`
Giroscopio	`gyroUpdateInterval`	`startGyroUpdatesToQueue:withHandler:`	`CMGyroHandler`
Magnetometro	`magnetometerUpdateInterval`	`startMagnetometerUpdatesToQueue:withHandler:`	`CMMagnetometerHandl`
Movimento del dispositivo	`deviceMotionUpdateInterval`	`startDeviceMotionUpdatesUsingReferenceFrame:` / `startDeviceMotionUpdatesUsingReferenceFrame:toQueue:withHandler:` / `startDeviceMotionUpdatesToQueue:withHandler:`	`CMDeviceMotionHandl`

Esempio

```
CMMotionManager *manager = [[CMMotionManager alloc] init];
[manager
        startAccelerometerUpdatesToQueue:[[NSOperationQueue alloc] init]
        withHandler:^(CMAccelerometerData *accelerometerData, NSError *error){
        /* processamento data block */
}];
```

Multimedia

Audio

iOs fornisce diversi framework che possono essere utilizzati nelle applicazioni per interagire con gli audio. Ognuno ha differenti funzionalità e la scelta sul quale utilizzare dipende dai tipo di operazioni che sono richieste all'interno dell'applicazione.

Framework	Utilizzo
Media Player	Riproduce suoni, audiolibri, audio e podcast dalla libreria iPod dell'utente
AV Foundation	Riproduce e registra audio utilizzando una semplice interfaccia Objective-C
Audio Toolbox	Riproduce audio con funzionalità di sincronizzazione, accede a pacchetti di audio in entrata, analizza flussi audio, converte formati audio e registra audio con accesso a singoli pacchetti
Audio Unit	Connette ed utilizza plug-in di elaborazione audio
OpenAL (1.1)	Fornisce la riproduzione audio posizionale nei giochi e in altre applicazioni

Accedere alla libreria iPod

L'accesso alla libreria iPod fornisce meccanismi per riprodurre suoni, audio libri e audio podcast; fornisce riproduzione di base, ricerca avanzata e controllo della riproduzione. Ci sono 2 modi per accedere agli oggetti multimediali:

- Il selettore multimediale (media picker), un view-controller che si comporta come l'interfaccia di selezione musicale dell'applicazione iPod built-in
- L'interfaccia di media query, supporta la specifica basata su predicato di elementi dalla libreria dell'iPod del dispositivo

Gli oggetti riperiti sono riprodotti utilizzando il riproduttore musicale `MPMusicPlayerController`.

Il framework Mediaplayer

Il framework del mediaplayer deve essere collegato al progetto ed importata mediante la direttiva di inclusione `#import <MediaPlayer/MediaPlayer.h>`. Sono presenti 2 tipi di riproduttori musicali:

- Il lettore musicale dell'applicazione riproduce la musica localmente all'interno della tua app
- Il lettore musicale iPod utilizza l'app iPod built-in

Un lettore musicale può essere istanziato con la linea di codice

```
MPMusicPlayerController *player = [MPMusicPlayerController applicationMusicPlayer];
```

Dopo che è stato creato una coda di riproduzione può essere impostata con i metodi

- `setQueueWithQuery:
- `setQueueWithItemCollection:`

Successivamente alla creazione i brani possono essere riprodotte con il metodo `play`.

Gestione della riproduzione

La classe `MPMusicPlayerController` è conforme al protocollo `MPMediaPlayback`, questo significa che il riproduttore implementa i metodi per cambiare lo stato della riproduzione:

- `play`
- `pause`
- `stop`
- altri metodi

e per gestire la coda di riproduzione:

- `skipToNextItem`
- `skipToBeginning`
- `skipToPreviousItem`

MPMediaItem

Un oggetto `MPMediaItem` rappresenta un singolo contenuto multimediale nella libreria iPod. Questo presenta un identificatore univoco, che persiste durante l'avvio delle applicazioni, un ambio raggio di metadati associati (titolo, artista, album) al cui è possibe accedere con il metodo `valueForProperty:`.

Sono presenti anche delle proprietà costati:

- `MPMediaItemPropertyTitle`
- `MPMediaItemPropertyAlbumTitle`
- `MPMediaItemPropertyArtist`
- `MPMediaItemPropertyAlbumArtist`
- `MPMediaItemPropertyGenre`

> Nota
> Ogni proprietà ha un tipo di ritorno differente. Per le istanze di `MPMediaItemPropertyTitle` il tipo di ritorno è `NSString`, mentre per le istanze di `MPMediaItemPropertyArtwork` il tipo di ritorno è un `MPMediaItemArtwork`

MPMediaQuery

Gli elementi multimediali nella libreria dell'iPod possono essere richiesti utilizzando la classe `MPMediaQuery`, e sono raccolti nelle collezioni `MPMediaItemCollection`. che possono essere acceduti attraverso la proprietà `collections`

MPMediaPickerController

Un `MPMediaPickerController` è un view-controller specializzato utilizzato per fornire un interfaccia grafica per selezionare elementi multimediali. Mostra a schermo una table-view con gli elementi che possono essere selezionati.

> Nota
> Toccando il pulsante "Fine" si chiuderà il selettore di elementi multimediali

Un `MPMediaPickerController` ha un delegato `MPMediaPickerControllerDElegate` che viene notificato quando un utente seleziona un elemento multimediale o cancella la selezione,. Questo avviene rispettivamente con i metodi:

- `mediaPicker:didPickMediaItems:`
- `mediaPickerDidCancel:`

Presentazione modale dei view-controller

Per presentare un view-controller modale, bisogna impostare la proprietà `modalTransitionStyle` e quindi il view-controller corrente dovrebbe chiamare `presentViewController:animated:completion`

```
MPMediaPickerController *picker = [[MPMediaPickerController alloc] init];
picker.modalTransitionStyle = UIModalTransitionStyleCoverVertical;
[self presentViewController:picker animated:YES completion:nil];
```

Il delegato del media-picker deve eliminare esplicitamente il selettore in `mediaPickerDidCancel:` con `dismissModalViewControllerAnimated:` (questo è automatico quando si tocca il pulsante "Fine")

Servizi audio di sistema

I servizi audio di sistema sono utilizzati per riprodurre effetti sonori dell'interfaccia utente o per invocare la vibrazione del dispositivo (se disponibile).

> Nota
> È richiesta l'inclusione `#import <AudioToolbox/AudioToolbox.h>`

Sono presenti dei vincoli che i file audio devono rispettare:

- le tracce devono avere durata ≤ 30 secondi
- devono avere un formato PCM lineare o IMA4 (IMA/ADPCM)
- devono avere estensione `.caf`, `.aif` o `.wav`

Il procedimento da seguire è:

1. Creare un oggetto sound-id con la funzione `AudioServicesCreateSystemSoundID()`
2. Riprodurre il suono con la funzione `AudioServicesPlaySystemSound()`

È possibile attivare la vibrazione con la seguente linea di codice

```
AudioServicesPlaySystemSound(kSystemSoundID_Vibrate);
```

AVAudioPlayer

La classe `AVAudioPlayer` fornisce una semplice interfaccia Objective-C per riprodurre i suoni.

> Nota
> È richiesta l'inclusione `#import <AVFoundation/AVFoundation.h>`

Apple consiglia di utilizzare questa classe per la riproduzione di audio che non richiedono posizionamento stereo o sincronizzazione precisa, e che non sia stato acquisito da un flusso di rete.

Questa classe può:

- Riprodurre audio di qualsiasi durata
- Riprodurre suoni da file o da buffer di memoria
- Riprodurre suoni a loop
- Riprodurre suoni multipli simultaneamente, però non con una precisa sincronizzazione
- Fornire controlli relativi al livello di riproduzione per ogni suono che è riprodotto
- Permette di cercare un punto particolare in un file audio, che supporta le funzionalità di avanzamento rapido e riavvolgimento
- Permette di ottenere dati sulla potenza audio che è possibile utilizzare per la misurazione del livello

Questa class ha un delegato `ACAudioPlayerDelegate` che prende le notifiche di interruzioni audio, errori di decodifica e completamento di riproduzione.

Esempio

```
/* get the URL of an audio file in the bundle */
NSString *soundFilePath = [[NSBundle mainBundle] pathForResource: @"sound" ofType: @"wav"];
NSURL *fileURL = [[NSURL alloc] initFileURLWithPath: soundFilePath];
/* create an AVAudioPlayer for the given audio file */
AVAudioPlayer *player = [[AVAudioPlayer alloc] initWithContentsOfURL:fileURL error:nil];
/* prepare the audio player for playback by preloading its buffers */
[player prepareToPlay];
/* set the delegate for the player */
[player setDelegate:self];
/* start the playback */
[player play];
```

AVAudioRecorder

La classe `AVAudioRecorder` fornisce una semplice interfaccia Objective-C per registrare suoni e permette di:

- Registrare finché l'utente non termina la registrazione
- Registrare per una specifica durata
- Mettere in pausa/proseguire una registrazione
- Prendere un input dati a livello audio che possono essere utilizzati per fornire la misurazione del livello

> Nota
> È richiesta l'inclusione `#import <AVFoundation/AVFoundation.h>`

La classe presenta un delegato `AVAudioRecorderDelegate` che viene notificato per di interruzioni audio, errori di decodifica e completamento di registrazione. L'utilizzo di un registratore d'audio è molto simile a quello di un riproduttore audio, a differenza che devono essere configurati numero di canali e bitrate.

> Nota
> Un registratore audio deve essere usato all'interno di un `AVAudioSession`, che è usato per impostare il contesto dell'audio per l'applicazione

Esempio

```
/* set the AVAudioSession parameters */
[[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryRecord error:nil];
[[AVAudioSession sharedInstance] setMode:AVAudioSessionModeVideoRecording error:nil];
[[AVAudioSession sharedInstance] setActive:YES error:nil];

/* get the URL of an audio file in the bundle */
NSString *soundFilePath = [[NSBundle mainBundle] pathForResource: @"sound" ofType: @"wav"];
NSURL *fileURL = [[NSURL alloc] initFileURLWithPath: soundFilePath];

/* get the URL for the recorded audio file */
NSURL *url = ...;

/* set the audio recorder parameters (format, quality, bitrate, number of channels, sample
rate) */
NSDictionary *settings = [NSDictionary dictionaryWithObjectsAndKeys:
                                          @(kAudioFormatMPEG4AAC), AVFormatIDKey,
                                          @(AVAudioQualityMax),
AVEncoderAudioQualityKey,
                                          @(16), AVEncoderBitRateKey,
                                          @(1), AVNumberOfChannelsKey,
                                          @(44100), AVSampleRateKey,
                                          nil];

/* create the audio recorder */
AVAudioRecorder *recorder = [[AVAudioRecorder alloc] initWithURL:url settings:settings
error:nil];

/* start recording */
[recorder record]; /* or use recordForDuration: */
```

Riproduzione video

Un oggetto `MPMoviePlayerController` gestisce la riproduzione di film da file o da flusso di rete, questo avviene in una view appartenente al riproduttore di film e prende posto sia a schermo intero sia in linea.

> Nota
> `MPMoviePlayerController` supporta la riproduzione di film wireless attraverso verso i dispositivi AirPlay

Sulla creazione viene inizializzato con l'URL del video da riprodurre e successivamente la vista creata può essere aggiunta ad una vista come sottovista.

Questa classe ha molte proprietà che possono essere usate per influenzare il suo comportamento:

- `allowsAirPlay`, specifica quando il riproduttore videro permette la riproduzione mediante AirPlay
- `fullscreen`, valore in sola lettura che indica se il video si trova a schermo intero. Il valore può essere cambiato con il metodo `setFullscreen:animated:`
- `controlStyle`, specifica lo stile dei controlli della riproduzione
- `initialPlaybackTime`, indica momento, in secondi, in cui il video deve partire
- `endPlaybackTime`, indica il momento, in secondi, in cui il video deve terminare
- `...`

Oltre ad avere svariate proprietà genera anche notifiche relativa allo stato della riproduzione del video:

- `MPMoviePlayerPlaybackStateDidChangeNotification`
- `MPMoviePlayerContentPreloadDidFinishNotification`
- `MPMoviePlayerDidEnterFullscreenNotification`
- `MPMoviePlayerDidExitFullscreenNotification`
- `...`

> Nota
> Per ricevere le notifiche è necessario registrarsi all'apposito avviso

Esempio

```
/* create a player to play a video at the given URL */
MPMoviePlayerController *player = [[MPMoviePlayerController alloc] initWithContentURL:url];
/* set the player's view frame */
[player.view setFrame:self.view.bounds];
/* add the player's view to the current view controller's view */
[self.view addSubview:player.view];
/* configure player's settings */
player.allowsAirPlay = YES;
player.initialPlaybackTime = 10;
player.endPlaybackTime = 20;
/* start playback */
[player play];
```

Scattare foto e video

Scattare foto e video è un procedimento abbastanza standard che richiede 3 passaggi:

1. Creare e presentare l'interfaccia della telecamera `UIImagePickerController`
2. L'utente scatta una foto/video o annulla
3. `UIImagePickerController` notifica il delegato rispetto al risultato dell'operazione

> Nota
> L'`UIImagePickerController` mostra un interfaccia con numerosi controlli

Prima di utilizzare il un oggetto di questa classe devono essere soddisfatte le seguenti condizioni:

- Il dispositivo deve avere una telecamera
- La telecamera del dispositivo deve essere disponibile per il tipo di sorgente dato controllando il valore di ritorno del metodo `isSourceTypeAvailable:`
- L'oggetto delegato, conforme ai protocolli `UIImagePickerControllerDelegate` e `UINavigationControllerDelegate`, deve essere implementato per rispondere alle iterazioni dell'utente con il controllore dell'image-picker.

Per specificare quando l'utente può prendere immagini, video o entrambi, deve essere impostata la proprietà `mediaTypes`. Questa proprietà consiste in un array non vuoto i cui elementi possono essere io valori costanti `kUTTypeImage` e `kUTTypeMovie`.

> Nota
> È necessario collegare il framework "Mobile Core Services" al progetto e inserire la direttiva `#import <MobileCoreServices/MobileCoreServices.h>`

Esempio

mostrare il controllore dell'image-picker

```
 /* assume the view controller conforms to UIImagePickerControllerDelegate and
UINavigationControllerDelegate */
- (BOOL)startCameraController{

        /* first check whether the camera is available for use */
        if (([UIImagePickerController
isSourceTypeAvailable:UIImagePickerControllerSourceTypeCamera] == NO))
                return NO;

        /* create an image picker controller and set its sourceType property */
        UIImagePickerController *camera = [[UIImagePickerController alloc] init];
        camera.sourceType = UIImagePickerControllerSourceTypeCamera;

        /* Displays a control that allows the user to choose picture or movie capture, if
both are available */
        camera.mediaTypes = [UIImagePickerController
        availableMediaTypesForSourceType:UIImagePickerControllerSourceTypeCamera];
```

```
        /* hide the controls for moving and scaling pictures */
        camera.allowsEditing = NO;

        /* set the delegate for the image picker controller */
        camera.delegate = self;

        /* present the image picker controller modally */
        camera.modalTransitionStyle = UIModalTransitionStyleCoverVertical;
        [self presentViewController:camera animated:YES completion:nil];

        return YES;
}
```

implementazione dei metodi delegati per l'image-picker

```
/* Responding to the user Cancel (dismiss the image picker controller) */
- (void) imagePickerControllerDidCancel:(UIImagePickerController *)picker{
        /* Dismiss the image picker controller */
        [picker.parentViewController dismissViewControllerAnimated:YES completion:nil];
}
/* Responding to the user user accepting a newly-captured picture or movie */
/* The info argument is a dictionary containing the original image and the edited image, if
an
image was picked; or a filesystem URL for the movie, if a movie was picked */
- (void) imagePickerController:(UIImagePickerController *)picker
                didFinishPickingMediaWithInfo:(NSDictionary *)info{

        NSString *mediaType = [info objectForKey:UIImagePickerControllerMediaType];

        /* Handle still image capture */
        if (CFStringCompare((CFStringRef) mediaType, kUTTypeImage, 0) == kCFCompareEqualTo){
                /* ... */
        }
        /* Handle video capture */
        if (CFStringCompare((CFStringRef) mediaType, kUTTypeMovie, 0) == kCFCompareEqualTo){
                /* ... */
        }
        /* Dismiss the image picker controller */
        [picker.parentViewController dismissViewControllerAnimated:YES completion:nil];
}
```

Prendere foto e video dalla libreria delle foto

Un oggetto di `UIImagePickerController` può essere usata anche per prendere oggetti dalla libreria delle foto del dispositivo. Gli step sono simili a quelli per l'acquisizione di contenuti multimediali, con alcune lievi differenze:

- Utilizzare l'album "Rullino fotografico" o l'album "Foto salvate" o l'intera libreria di foto come sorgente multimediale
- L''utente sceglie i media salvati in precedenza invece di acquisire nuovi media

Per configurare il selettore per sfogliare i media salvati, bisogna impostare la proprietà `sourceType` su:

- `UIImagePickerControllerSourceTypePhotoLibrary`, per avere accesso all'album del dispositivo includendo anche il rullino fotografico
- `UIImagePickerControllerSourceTypeSavedPhotosAlbum`, per avere accesso solo al rullino fotografico