PuntoMovimientoSonido

Aplicación iOS creada por Daniel Barros López y Yoji Sargent Harada

Requerimientos

Se pedirá una appMovimientoSonido que reconozca un patrón de movimientos de vuestra elección usando el giroscopio y/o el acelerómetro, una vez detectado el patrón, se reproducirá un sonido.

Tutoriales

En el desarrollo de esta aplicación aprendimos a manejar parte de la librería CoreMotion para obtener información del acelerómetro, a reproducir sonidos y hacer vibrar a nuestro dispositivo. También creamos una aplicación para Apple Watch con comunicación mediante mensajes con la aplicación principal (detallamos lo aprendido de watchOS en un tutorial a parte).

1. Detección de un patrón de movimientos

Primero importamos el framework `CoreMotion` y creamos un objeto `CMMotionManager` mediante el cual obtendremos información del acelerómetro.

```
let manager = CMMotionManager()
```

Para empezar a recibir esta información especificamos cada cuanto queremos que se actualize mediante la propiedad `deviceMotionUpdateInterval`. Seguidamente llamamos al método `startDeviceMotionUpdates()`

```
if manager.deviceMotionAvailable {
      manager.deviceMotionUpdateInterval =
Constants.gestureRecognitionUpdateInterval
      manager.startDeviceMotionUpdates()
}
```

Una vez hecho esto podemos acceder a la propiedad `deviceMotion` de nuestro manager y obtener información tal como la aceleración de nuestro dispositivo (`userAcceleration`). Esto lo hacemos dentro de una función que se ha de llamar con la misma frecuencia con la que se actualiza la información en `deviceMotion`, lo que logramos mediante un temporizador.

```
timer = NSTimer(timeInterval: Constants.gestureRecognitionUpdateInterval,
target: self, selector: "detectPattern", userInfo: nil, repeats: true)
NSRunLoop.mainRunLoop().addTimer(timer!, forMode: NSDefaultRunLoopMode)
```

Dentro de nuestra función `detectPattern()` accedemos a la aceleración en el eje x (movimientos hacia la derecha o izquierda)

```
let xAcc = manager.deviceMotion?.userAcceleration.x
```

y también a la orientación del dispositivo (`FaceUp, FaceDown, Portrait`, etc).

```
let orientation = UIDevice.currentDevice().orientation
```

Con esto y guardando el paso en el que nos encontramos detectamos un patrón de movimientos concreto: empezando con el dispositivo mirando hacia arriba lo movemos a la derecha, le damos la vuelta, lo movemos a la izquierda y le volvemos a dar la vuelta:

```
func detectPattern() {
        guard let xAcc = manager.deviceMotion?.userAcceleration.x else {
            return
        }

        switch gesturePhase {
        case .Beginning:
            if UIDevice.currentDevice().orientation == .FaceUp && xAcc >
Constants.gestureMinXAcceleration {
                gesturePhase = .MovingRight
            }
        case .MovingRight:
            if UIDevice.currentDevice().orientation == .FaceDown && xAcc < -
Constants.gestureMinXAcceleration {
                gesturePhase = .MovingLeft
            }
        case .MovingLeft:
            if UIDevice.currentDevice().orientation == .FaceUp {
                gesturePhase = .Beginning
                patternDetected()
            }
        }
}
```

Debemos recordar invalidar el temporizador y parar la actualización de información del acelerómetro en algún momento (en el destructor de nuestro view controller por ejemplo)

```
timer?.invalidate()
manager.stopDeviceMotionUpdates()
```

Como último, hay que notar que la orientación del dispositivo tal y como la obtenemos, a través de la clase `UIDevice`, cambia con un pequeño retardo. Esto no supone un problema para la detección de nuestro patrón, pero para otros casos podría ser conveniente obtener esta información por otros medios.

2. Reproducción de sonidos

Primero importamos el framework `AVFoundation` y creamos un objeto `AVAudioPlayer`. Es importante que sea una propiedad de nuestro view controller ya que este es el objeto encargado de reproducir el sonido, y lo hará asíncronamente. Si lo declaramos dentro de una función se liberará al salir del scope y el sonido no llegará a reproducirse.

```
var player: AVAudioPlayer!
```

Inicializamos el objeto con una URL al archivo de audio que queramos. Este archivo debemos incluirlo dentro de la carpeta que contiene los archivos de código (no en la carpeta padre que contiene el archivo del proyecto)

```
let url = NSURL(fileURLWithPath:
NSBundle.mainBundle().pathForResource("buttonIn.mp3", ofType: nil)!)
player = try! AVAudioPlayer(contentsOfURL: url)
```

Una vez hecho esto ya podemos reproducir nuestro sonido cuando queramos llamando a `prepareToPlay()` seguido de `play()`

```
player.prepareToPlay()
player.play()
```

3. Vibración

Con el framework `AVFoundation` importado, usamos esta función:

```
AudioServicesPlayAlertSound(SystemSoundID(kSystemSoundID_Vibrate))
```