PuntoGestosFoto

Aplicación iOS creada por Daniel Barros López y Yoji Sargent Harada

Requerimientos

Se pedirá una appGestosFoto que reconozca un patrón de gestos sobre pantalla de vuestra elección. Una vez detectado el patrón se debe ejecutar la aplicación de cámara y realizar automáticamente una foto transcurridos 3 segundos. Los gestos a detectar son similares a los que se realizan con la pantalla de bloqueo Android, aquí podéis ver algunos ejemplos, cuanto más complejo sea el que se detecta mejor puntuación tendrá.

Tutoriales

En el desarrollo de esta aplicación hemos aprendido a reconocer cuando un dedo toca una vista (`UIView`) de nuestra interfaz y a obtener información relativa a este evento, además de cómo acceder a la cámara de nuestro dispositivo mediante la clase `UIImagePickerController`.

1. Reconocimiento de un toque

Para manejar eventos relacionados con toques debemos implementar los siguientes cuatro métodos (mediante override) en nuestra subclase de `UIView`

```
// Llamado en el primer momento en el que el dedo toca la pantalla (y nuestra
view)
override func touchesBegan(touches: Set<UITouch>, withEvent event: UIEvent?)
{
}

// Llamado cuando el dedo se ha movido (dentro de nuestra view)
override func touchesMoved(touches: Set<UITouch>, withEvent event: UIEvent?)
{
}

// Llamado cuando el dedo se ha separado de la pantalla (justo antes estaba
sobre nuestra view)
override func touchesEnded(touches: Set<UITouch>, withEvent event: UIEvent?)
{
}

// Llamado cuando algún evento ha causado que se deje de detectar nuestro
dedo en la view (bloqueo del dispositivo, recibimos una llamada, etc.)
override func touchesCancelled(touches: Set<UITouch>?, withEvent event:
UIEvent?)
{
}
```

Si decidimos ignorar el evento pero queremos que sea manejado por una vista padre, debemos llamar a la implementación del mismo método dada por `UIView`. Por ejemplo:

```
super.touchesBegan(touches, withEvent:event)
```

Dentro de cualquiera de estas funciones podemos acceder a la información contenida en los objetos de tipo `UITouch` contenidos en el conjunto touches. Por ejemplo, en nuestra aplicación obtenemos la localización dentro de la vista de nuestro toque, para poder averiguar que subsección hemos tocado y hacer toda la lógica relacionada con nuestro patrón. Lo conseguimos usando el método `locationInView(_:)` de `UITouch`.

```
let location = touches.first!.locationInView(self)
```

2. Sacar una foto

Para acceder a la cámara de nuestro dispositivo usamos un objeto `UIImagePickerController`. Podemos configurarlo a través de sus propiedades, siendo las más importantes `sourceType`, que indica la fuente de la que queremos obtener la imagen, y `cameraOverlayView`, que nos permitirá superponer nuestra propia UI a la de la cámara.

```
var picker = UIImagePickerController()
if UIImagePickerController.isSourceTypeAvailable(.Camera) {
      picker.delegate = self
      picker.sourceType = .Camera
      picker.showsCameraControls = false
      picker.cameraOverlayView = overlayView
}
```

Aquí `overlayView` no es más que una `UIView` cualquiera que hayamos creado y que se añadirá a la jerarquía de vistas del objeto `UIImagePickerController`. En nuestra aplicación creamos una vista con un label que muestra el tiempo que queda antes de que se tome la foto y un botón para cancelar.

El `delegate` de nuestro `picker` puede implementar varias funciones relacionadas con los eventos que se pueden dar durante el proceso de la toma de una foto. El más importante es `imagePickerController(_: didFinishPickingMediaWithInfo:)`, que se llamará cuando la fotografía se ha tomado con éxito.

```
func imagePickerController(picker: UIImagePickerController, didFinishPickingMediaWithInfo
info: [String : AnyObject]) {

      let image = info[UIImagePickerControllerOriginalImage] as? UIImage
}
```

Como vemos arriba, podemos obtener la imagen mediante el diccionario `info`. Se encuentra almacenada con la clave `UIImagePickerControllerOriginalImage`.