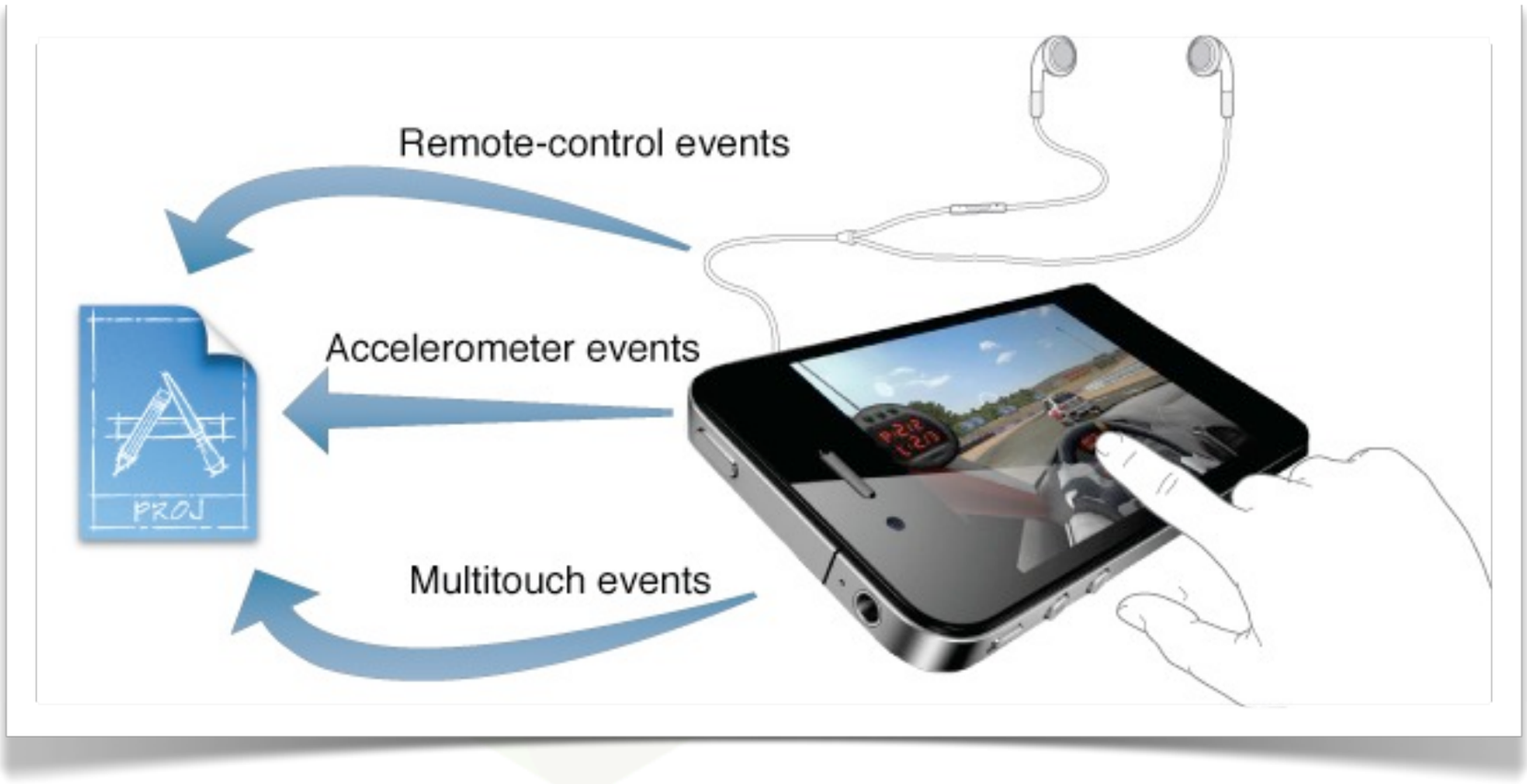
Tema 4

Eventos

¿Qué es un evento?



Es un objeto enviado a una aplicación para informarla sobre las acciones del usuario.

Tipos de Eventos

- **Eventos Multitoque.** Se lanzan cuando el usuario toca la una vista. Hay muchos tipos de toques y combinaciones. Estudiaremos los **pinch**, **pan**, **swipe** y muchos otros.

- **Eventos del Acelerómetro.** Detectan cualquier cambio de orientación y posición del dispositivo físico. Se utilizan para situar al usuario en los mapas (brújula) y sobre todo para los juegos.

- **Eventos de Control Remoto.** Se lanzan cuando el usuario manipula los controles de los auriculares (subir y bajar volumen y reproducir/ pausar) o a través de accesorios externos (docks).

Clase UIEvent

- Clase del **UIKit Framework** utilizada para representar un evento. Es la clase que recoge todos los tipos de eventos.

- Encapsula la información relativa al evento (por ejemplo los *touches* asociados).

- 3 tipos: *UIEventTypeTouches, UIEventTypeMotion y UIEventTypeRemoteControl*.

Conceptos

- ***Gesture*.** Es una secuencia de eventos que ocurren desde que se pulsa la pantalla con uno o más dedos hasta que se deja de pulsar la pantalla.

- ***Touch*.** Es cuando un dedo se encuentra en la pantalla.

- ***Tap*.** Tiene lugar cuando se pulsa con un dedo e inmediatamente se suelta, sin realizar ningún movimiento.

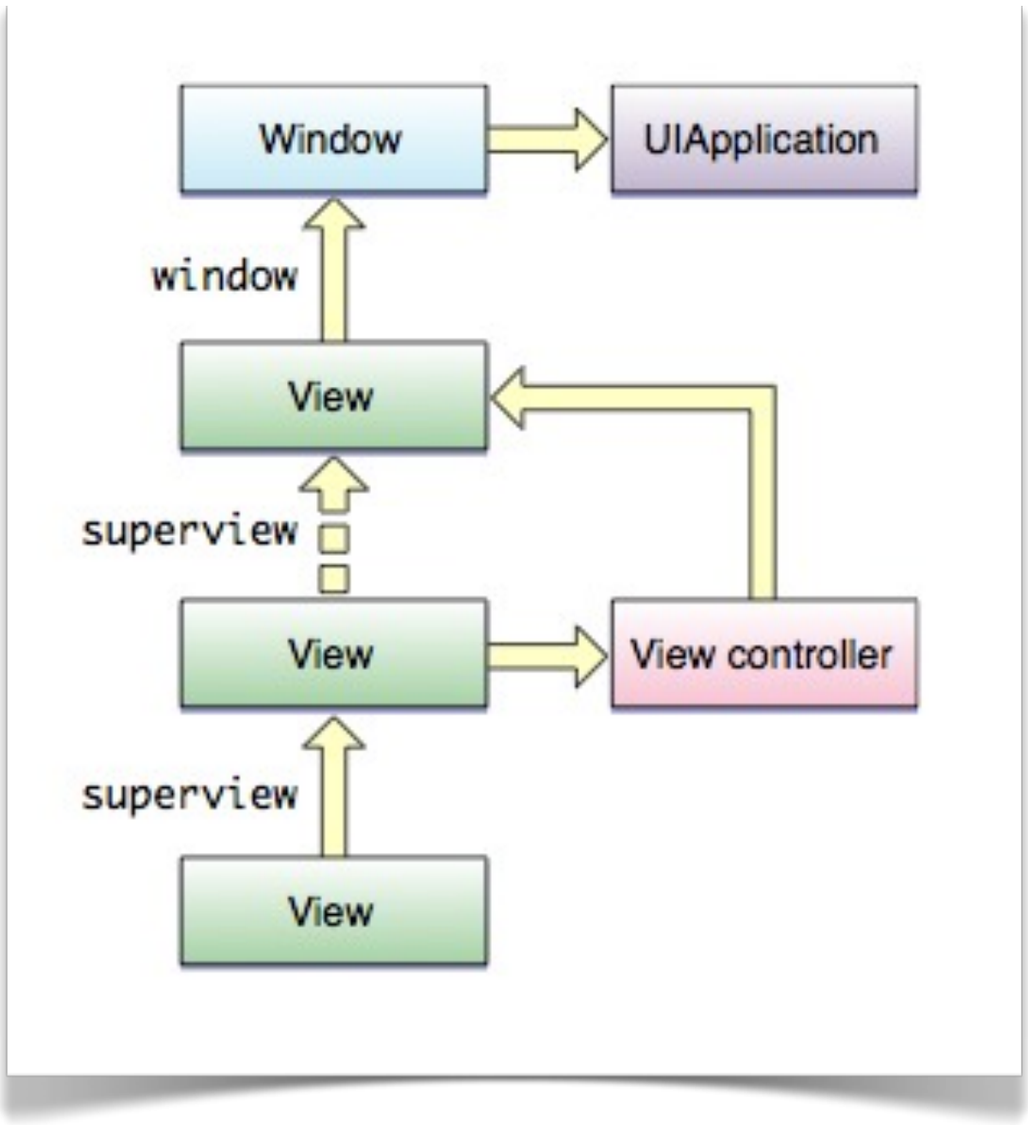
¿Quién responde al evento?

- Existe una cadena de “respondedores” que deben contenter un objeto ***UIResponder***.

- El primer objeto que responde es el que está en interacción con el usuario.

- *UIView* es una subclase de ***UIResponder*** y ***UIControl*** es subclase de *UIView*, con lo que todos responden al evento.

Cadena de Respondedores en iOS



Manejadores

- Si un objeto no tiene una rutina que maneje el evento, se pasa al siguiente objeto en la cadena.
- Normalmente si un objeto responde, éste es el último en hacerlo.
- No obstante, se puede redirigir el evento a los respondedores siguientes.

Re-dirigiendo eventos

- Ejemplo de pseudo-código.
  Redirección de un evento, según cumpla o no la condición.

```
-(void)respondToFictionalEvent:(UIEvent *)event {

   if (someCondition)
        [self handleEvent:event];
   else
        [self.nextResponder respondToFictionalEvent:event];

}
```

Métodos de Notificación 1/2

- Al sobreescribir **touchesBegan** capturamos el inicio de la interacción.

```
- (void) touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
    // When one or more fingers touch down the screen.
    NSUInteger numTaps = [[touches anyObject] tapCount];
    NSUInteger numTouches = [touches count];

}
```

Métodos de Notificación 2/2

- Otros métodos comunes son **touchesMoved**, **touchesEnded** y **touchesCancelled**.

```
- (void) touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event
{
   // Cuando se mueven uno o más dedos.
}
```

```
- (void) touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event
{
   // Cuando uno o más dedos se levantan de la pantalla.
}
```

```
- (void) touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event
{
   // Cuando se cancela la secuencia de toques debido a un evento de sistema.
}
```

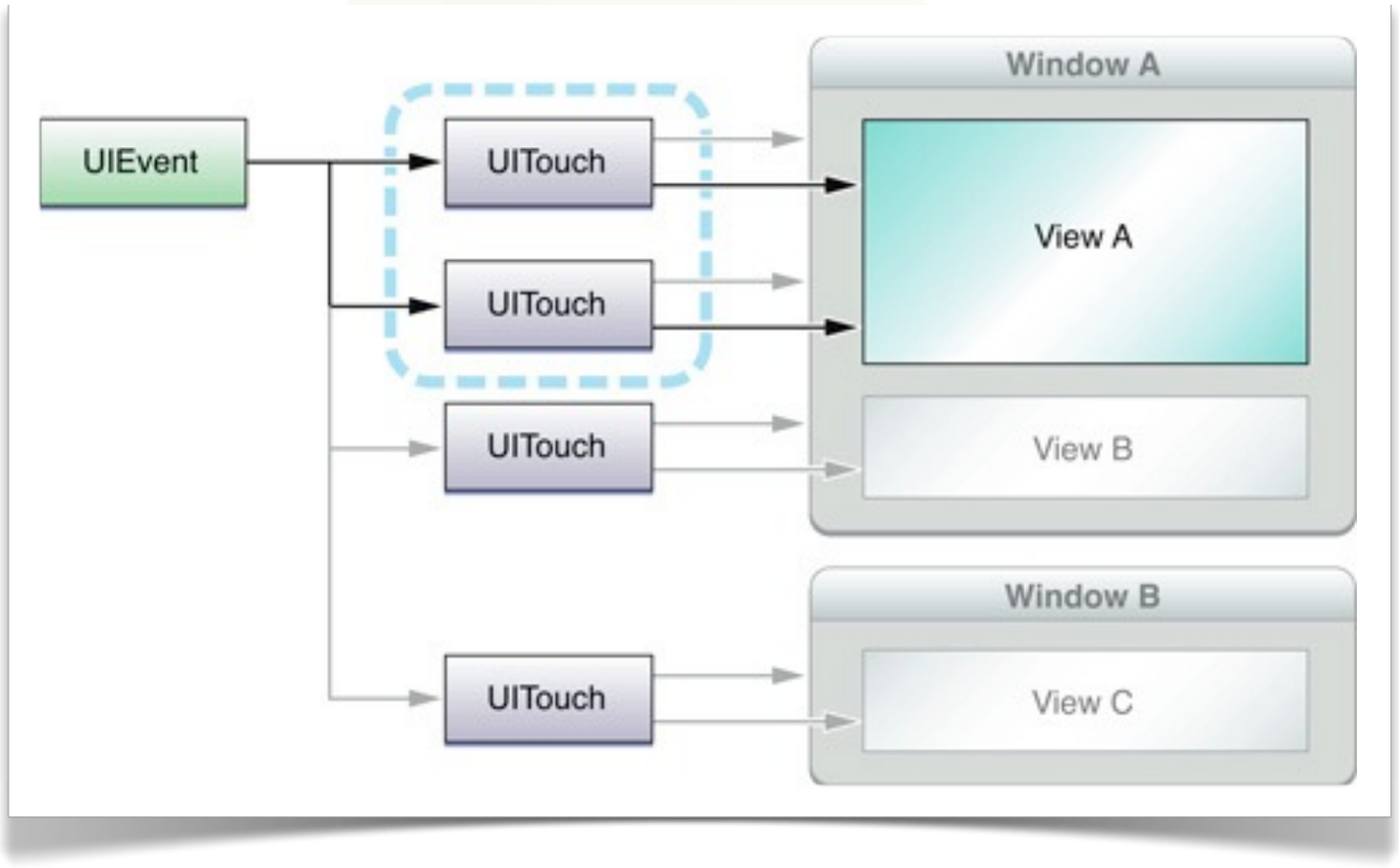
Utilidades

- Mediante **locationInView** podemos obtener la posición exacta del *touch*.

```
UITouch *touch = [touches anyObject];
CGPoint location = [touch locationInView: [self view]];
```

Utilidades

- Todos los ***touch*** que pertenecen a la vista.



Detectando Múltiples *Taps*

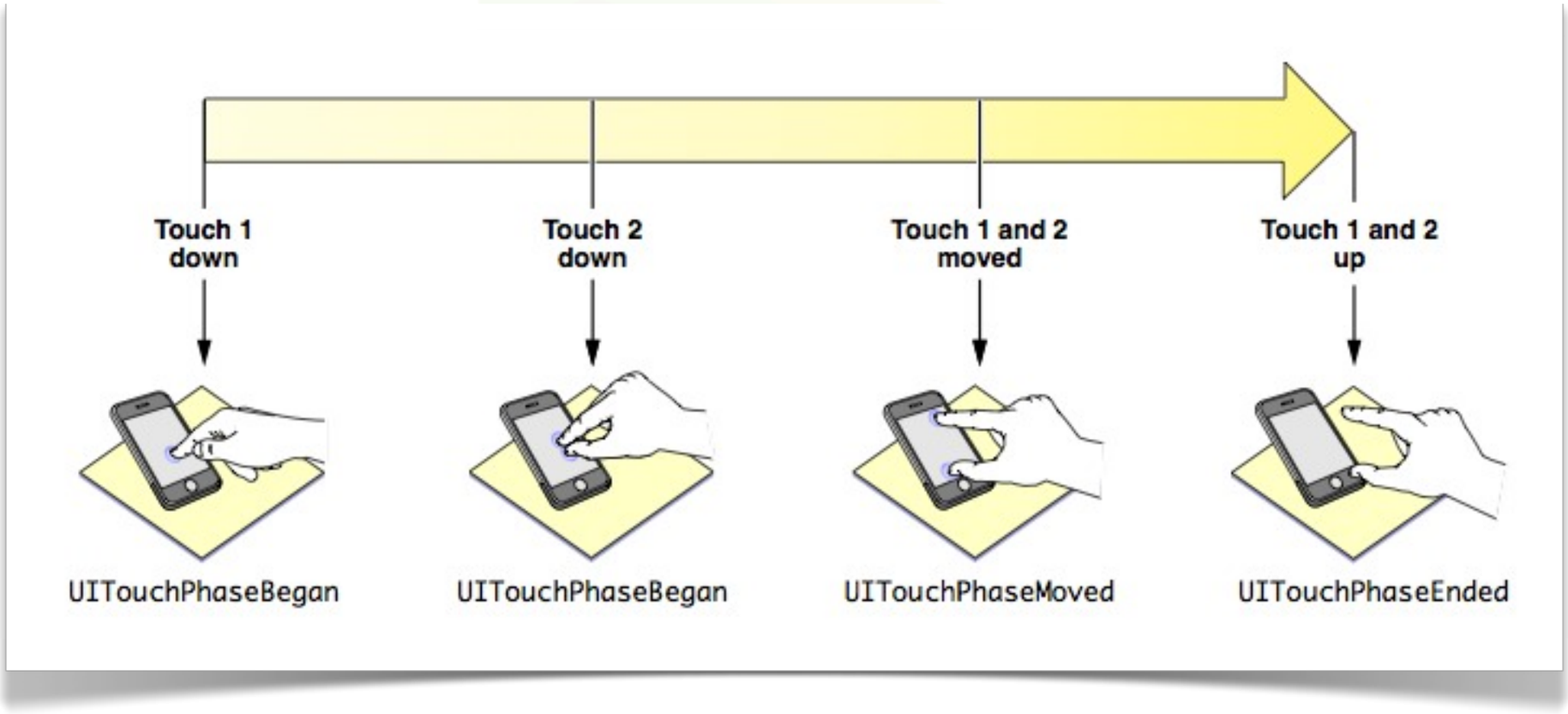
- La cuenta de los ***taps*** es accesible desde cada ***touch*** (UITouch).

- Para ello, debemos utilizar el método ***tapCount*** y su resultado es el número de ***taps*** realizados.

```
UITouch *touch = [touches anyObject];
NSUInteger tapCount = [touch tapCount];
```

Eventos *Multitouch*

- Implica el lanzamiento de varios tipos de eventos por parte del usuario.



Swipes y Drags

- Para registrar un desplazamiento (***swipe***) se debe seguir la acción del usuario sobre un eje determinado.

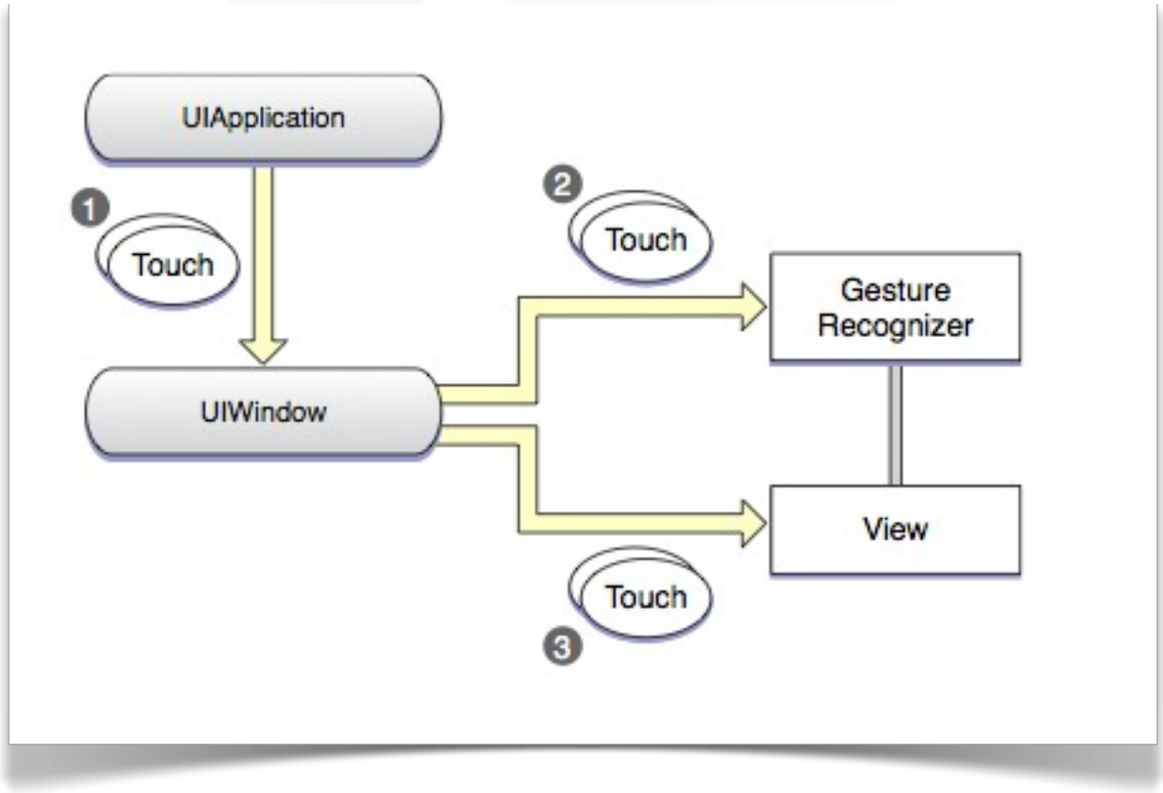
- La detección del arrastre (***drag***) implica la pulsación de un objeto y su desplazamiento dentro de la vista.

Tipos de Gestos

GESTO	Clase UIKit
Tapping (cualquier número de toques)	**UITapGestureRecognizer**
Pinching (para hacer zoom)	**UIPinchGestureRecognize**
Dragging (arrastrar)	**UIPanGestureRecognize**
Swiping (deslizar en cualquier dirección)	**UISwipeGestureRecognize**
Rotating (los dedos se mueven en direcciones opuestas)	**UIRotationGestureRecognize**
Long press (tocar y mantener)	**UILongPressGestureRecognize**

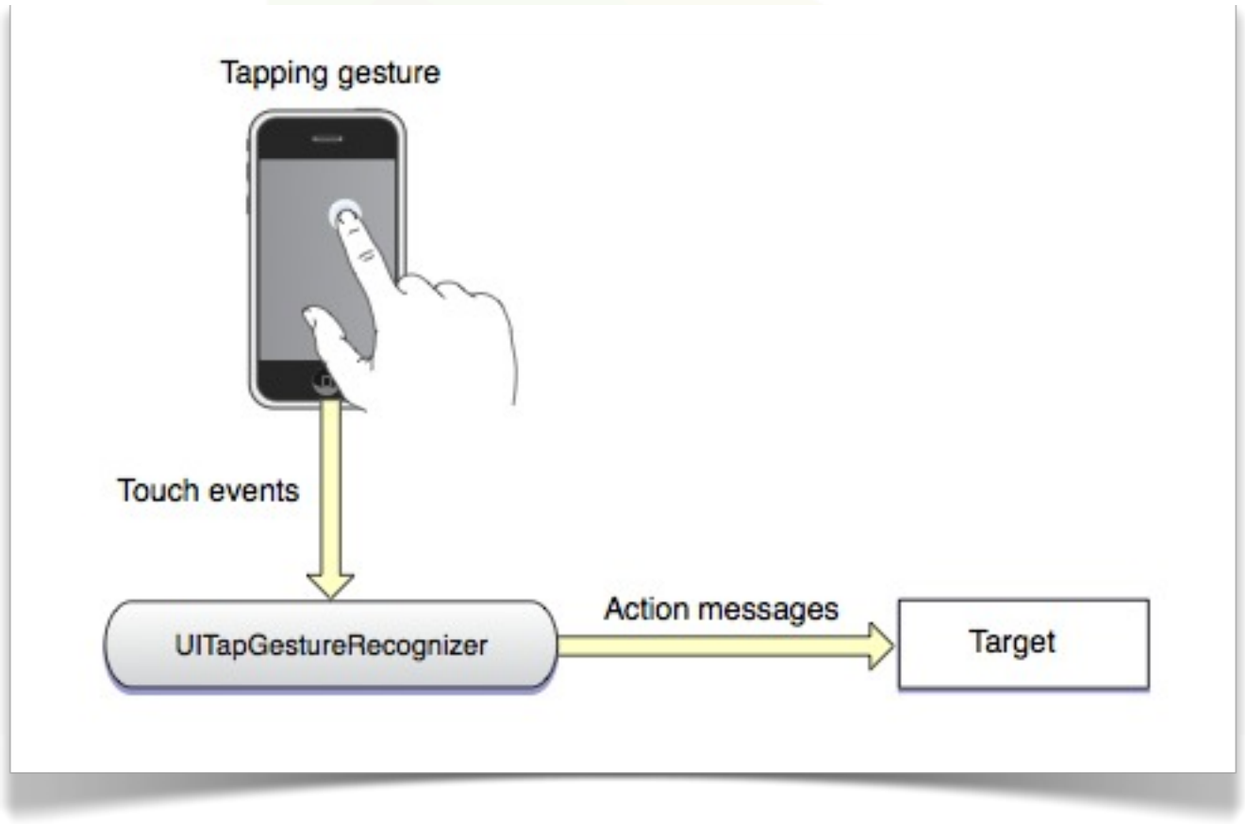
Reconocedor de gestos

- Es necesario añadir a la vista un reconocedor de gestos.



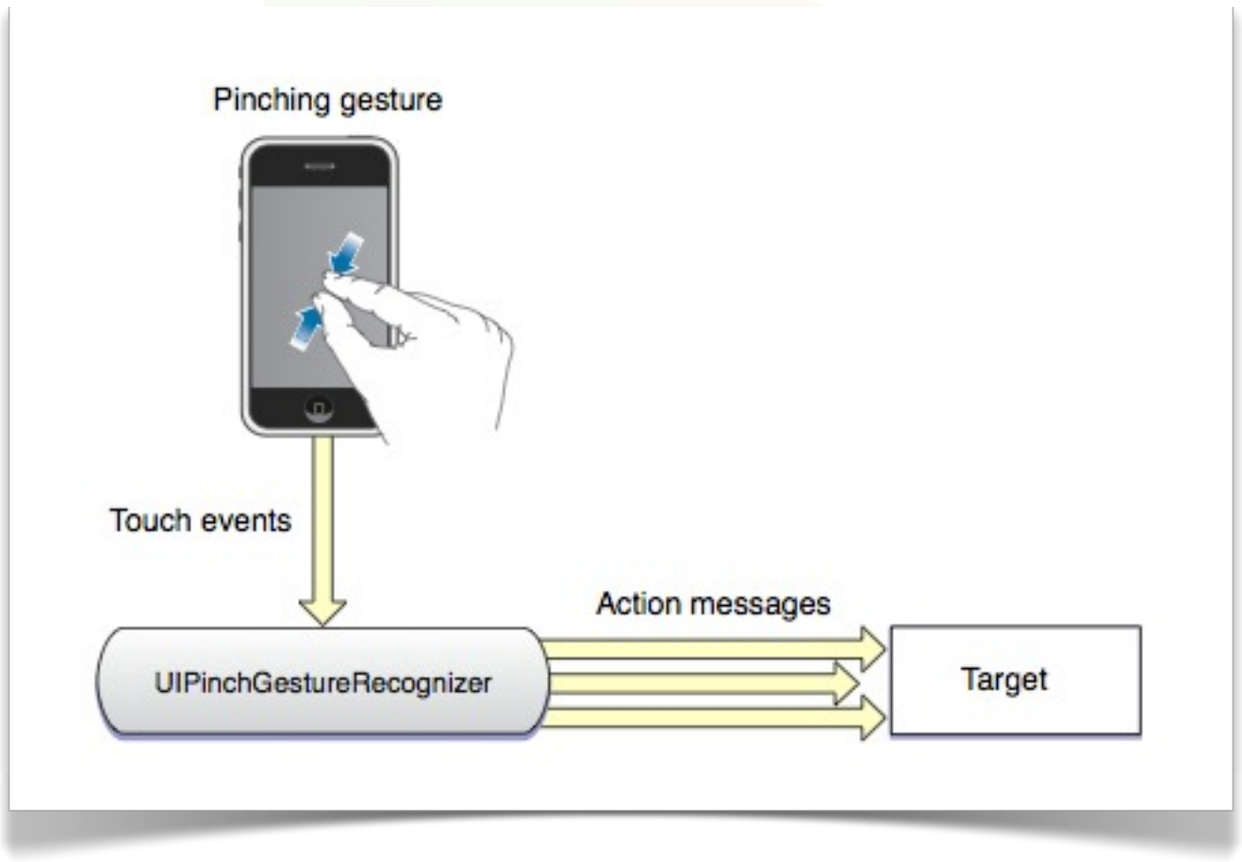
Gestos discretos

- Un gesto discreto lanza un solo evento.



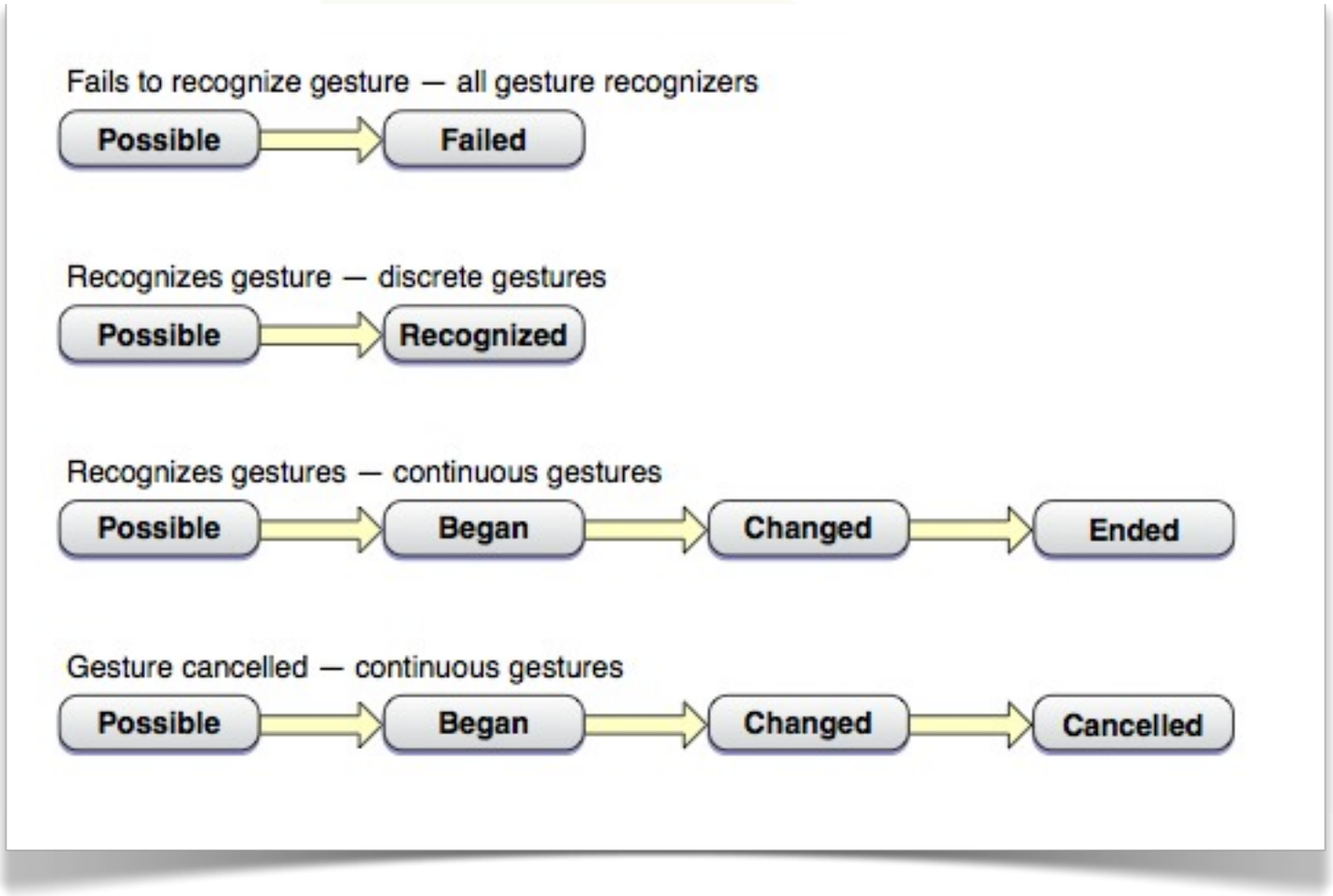
Gestos continuos

- Un evento continuo lanza varios eventos.



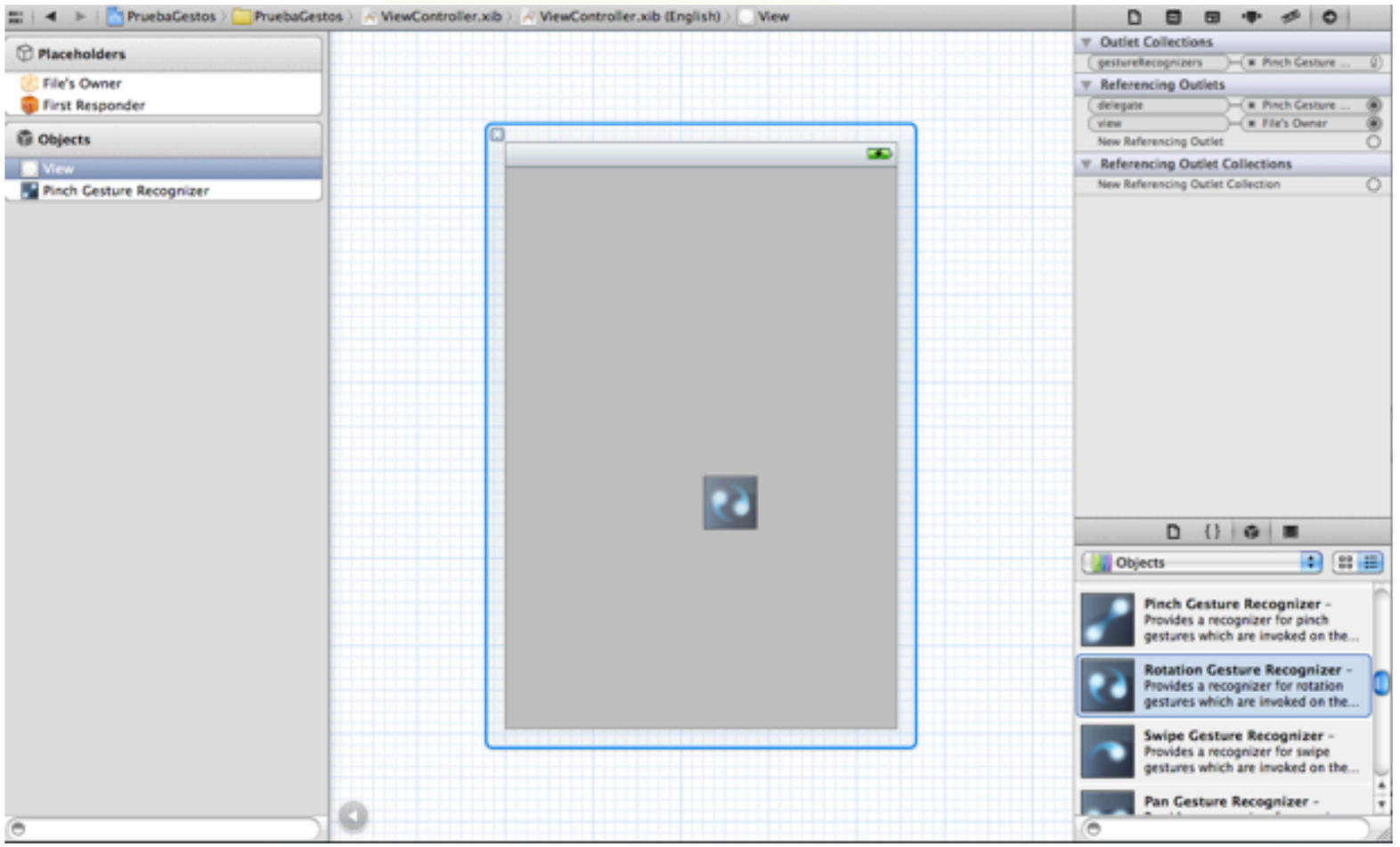
Posibles estados

- El reconocedor de gestos presenta los siguientes diagramas de flujo.



Gestos mediante Interface Builder

- Podemos añadir los manejadores para los gestos mediante Interface Builder.

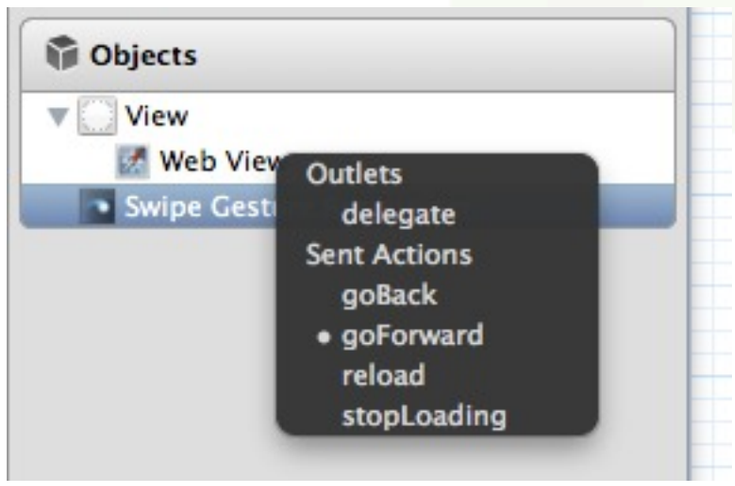


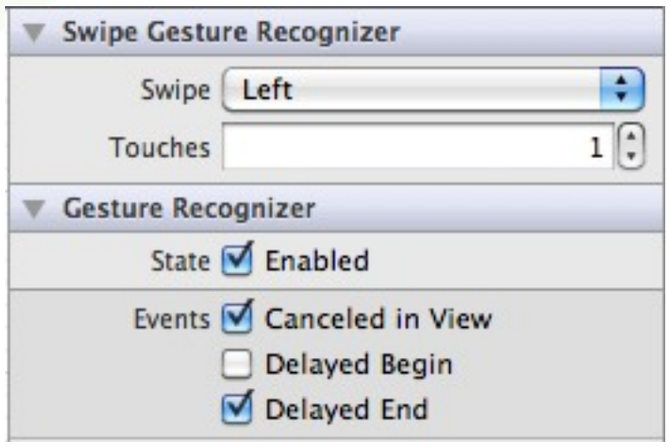
Reconocedor de Gestos en IB

- Se añaden a la vista o al objeto que deseemos que reconozca.

- Se configuran las propiedades del gesto.

- Se asocia al método que deseemos ejecutar mediante ctrl+click.





Pinching en el Simulador

- Para realizar el gesto de ***pinching*** en el simulador, debes mantener la tecla **alt** pulsada, a la vez que haces click en el ratón.

- Posteriormente, desplaza el ratón en la dirección que desees hacer el gesto.

Shaking-Motion

- Cuando el usuario sacude el dispositivo, el sistema evalúa la información del acelerómetro y si sigue cierto criterio, se interpreta con un gesto de sacudida (***shake***).

- Es una acción discreta. El sistema avisa cuando comienza y termina y no proporciona eventos intermedios.

Obteniendo la orientación del dispositivo

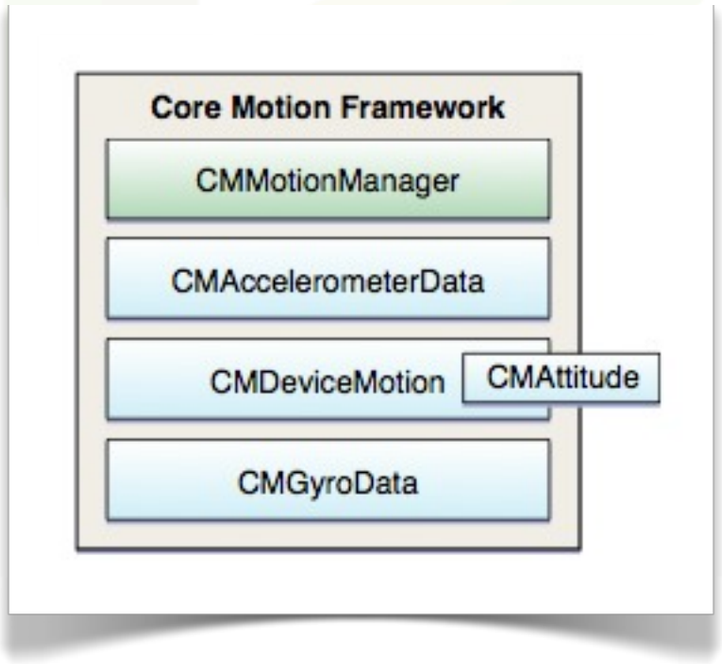
- Se debe utilizar la clase **UIDevice** e invocar al método **beginGeneratingDeviceOrientationNotifications.**

- A partir de ese momento, obtendremos la información de la orientación mediante el atributo *orientation*. Si se registra **UIDeviceOrientationDidChangeNotification** se pueden obtener las notificaciones de cambio de orientación.

- Cuando no se necesitan registrar los cambios de orientación, debemos invocar al método **endGeneratingDeviceOrientationNotifications**.

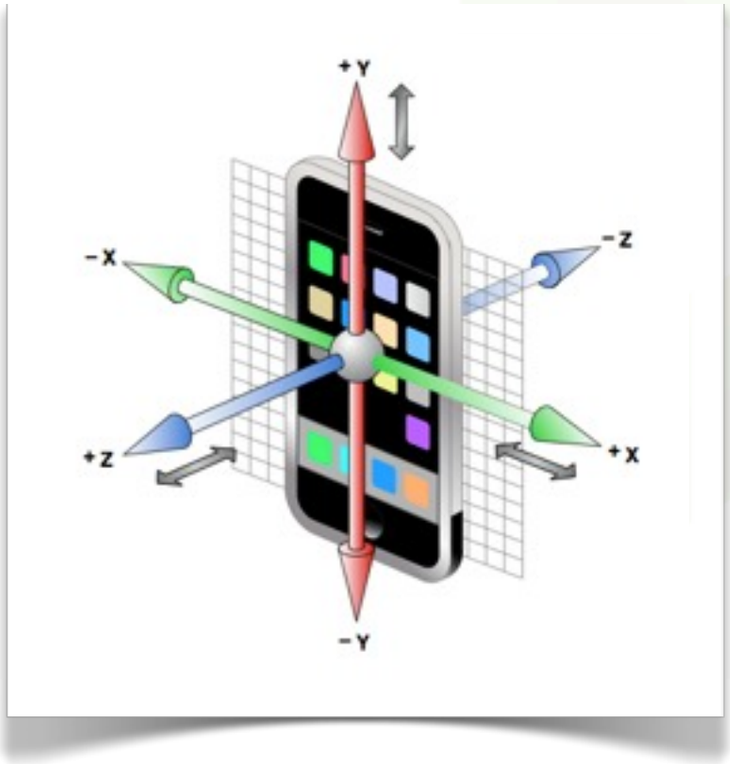
Core Motion

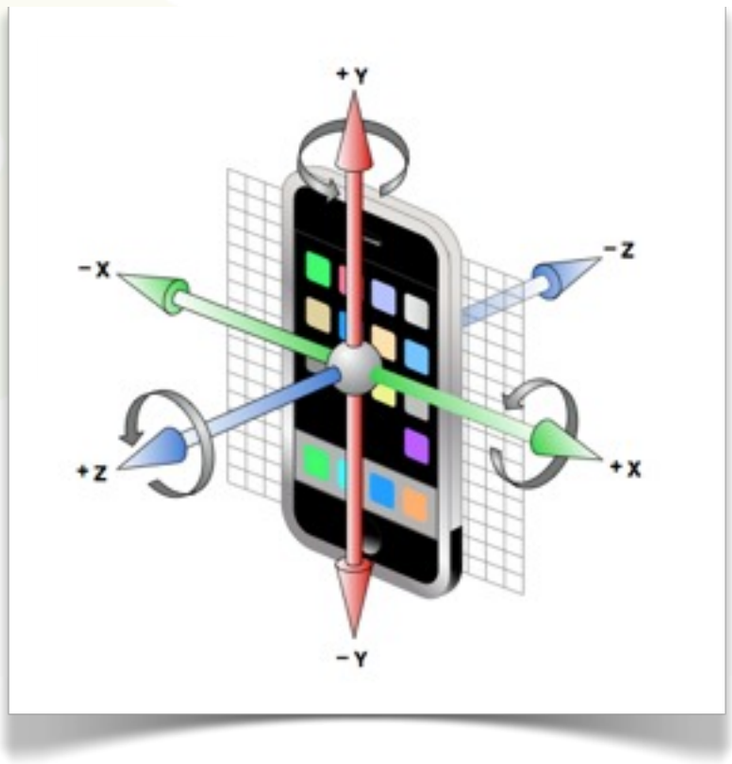
- Es un *framework* del sistema utilizado para recibir información de los sensores del dispositivo.
- Incluyen el acelerómetro y el giroscopio.



Core Motion

- Tipos de movimientos registrados.





Eventos de Movimiento

- El objeto **CMAccelerometerData** encapsula una estructura que captura la aceleración sobre cada uno de los ejes espaciales.

- El objeto **CMGyroData** registra la velocidad de rotación sobre los ejes espaciales.

- El objeto **CMDeviceMotion** encapsula varias medidas, incluyendo la postura y medidas más útiles de la rotación y aceleración.

Control Remoto

- La Vista o el controlador de Vista deben ser el ‘primer respondedor’.

- Para ello debe sobreescribir **canBecomeFirstResponder** *(*de **UIResponder***)* y devolver YES.

```
- (void)viewWillAppear:(BOOL)animated {
    [super viewWillAppear:animated];
    [[UIApplication sharedApplication] beginReceivingRemoteControlEvents];
    [self becomeFirstResponder];
}
```

Desactivando el Control Remoto

- Cuando la vista o controlador ya no van a manejar el audio o el vídeo, se debe desactivar.

- Adicionalmente, se debe anular la orden de ser el primer receptor.

```
- (void)viewWillDisappear:(BOOL)animated {
    [super viewWillDisappear:animated];
    [[UIApplication sharedApplication] endReceivingRemoteControlEvents];
    [self resignFirstResponder];
}
```

Manejando los eventos remotos

```
- (void)remoteControlReceivedWithEvent:(UIEvent)theEvent {
    AVAudioPlayer *player = self.audioPlayer; // initialized with sound file
    if (theEvent.type == UIEventTypeRemoteControl) {
          switch(theEvent.subtype) {
                  case UIEventSubtypeRemoteControlPlay:
                          BOOL success=YES;
                          success = [player play];
                          if (!success)
                                  return;
                          break;
                  case UIEventSubtypeRemoteControlPause:
                          [player pause];
                          break;
                  case UIEventSubtypeRemoteControlStop:
                          [player stop];
                          [player.currentTime = 0];
                          break;
                  default:
                          return;
          }
    }
}
```