Desarrollo de Apps para iOS
Gestos

IWEB,LSWC 2013-2014
Santiago Pavón

ver: 2013.11.05 p1

Terminología

- **Gesto**: Secuencia de eventos provocada por varios dedos, y que termina al retirar los dedos, o cuando el sistema lo interrumpe.
- **Evento**: generado al interaccionar con la pantalla, e informa de los toques ocurridos.
- **Toque**: representa el contacto de un dedo en la pantalla.

UIResponder

- **UIResponder**:
  - Responde y maneja eventos.
    - de tipo touch, motion, de dispositivos de control remoto.
  - Es la **superclase** de `UIView`, `UIControl`, `UIApplication` `UIViewController`, etc.

- Algunos métodos de `UIResponder`:

```
- (void) touchesBegan:(NSSet *)touches
              withEvent:(UIEvent *)event
- (void) touchesMoved:(NSSet *)touches
              withEvent:(UIEvent *)event
- (void) touchesEnded:(NSSet *)touches
              withEvent:(UIEvent *)event
- (void) touchesCancelled:(NSSet *)touches
                  withEvent:(UIEvent *)event
```

Responder Chain

- **First Responder**:
  - objeto con el que estamos interaccionando.
  - es el comienzo de la cadena de respuesta.
- Un evento avanza por la cadena de respuesta hasta llegar a `UIWindow`, luego a `UIApplication`, y finalmente se descarta.
  - El avance por la cadena suele romperse cuando el evento es atendido en algún punto intermedio.

Demo: Doble Tap

- Proyecto: Movimiento Relativo
  - **MovilesView** es una UIView donde se redefine el método **touchesBegan:withEvent:**.
  - Un toque doble sitúa la cámara en el móvil más cercano.

```
- (void) touchesBegan:(NSSet *)touches
           withEvent:(UIEvent *)event {

    UITouch * touch = [touches anyObject];
    int taps = [touch tapCount];
    if (taps == 2) {
        CGPoint p = [touch locationInView:self];

        int h = self.bounds.size.height;
        int w = self.bounds.size.width;
        NSLog(@"Doble tap en %f,%f",p.x-w/2,h/2-p.y);
        Position * pos = [Position dequeuePosition];
        [pos setX:p.x-w/2 andY:h/2-p.y andA:0];
        if (self.camera)
            [pos setPosicion:pos
                placedAtBase:self.camera.posRaiz];
        Sistema * sist = [Sistema sistema];
        Movil * m = [sist nearestMovil:pos];
        NSLog(@"Movil mas cercano en %@",m.nombre);
        self.camera = m;
    }
}
```

Demo: Programar un Swipe

- Hay que sobreescribir los métodos:
  - **`-touchesBegan:withEvent:`**
    - Guardamos la posición inicial
  - **`-touchedMoved:withEvent:`**
    - Hay que comprobar que:
      - la posición actual no se ha desviado mucho vertical u horizontalmente.
      - ya se ha recorrido una distancia mínima para aceptar el gesto.

(la demo no mira la trayectoria intermedia)

```
#define kMinLength  30
#define kMaxError   5

CGPoint  initialPoint;

- (void)touchesBegan:(NSSet*)touches withEvent:(UIEvent*)event
{
    UITouch *touch = [touches anyObject];
    initialPoint = [touch locationInView:self.view];
}

- (void)touchesMoved:(NSSet*)touches withEvent:(UIEvent*)event
{

    UITouch *touch = [touches anyObject];
    CGPoint currentPoint = [touch locationInView:self.view];

    CGFloat diffX = fabsf(initialPoint.x - currentPoint.x);
    CGFloat diffY = fabsf(initialPoint.y - currentPoint.y);

    if (diffX >= kMinLength && diffY <= kMaxError) {
        // Detectado SWIPE Horizontal - Hacer algo
    } else if (diffY >= kMinLength && diffX <= kMaxError) {
        // Detectado SWIPE Vertical - Hacer algo
    }
}
```

Reconocedores de Gestos

Reconocedores de Gestos

- Un reconocedor de gestos es un objeto que vigila los eventos que ocurren en una determinada view, y cuando reconoce el gesto (o parte del gesto) que lo caracteriza, ejecuta las targets-actions que tiene configuradas.
- Reconocedores de gestos predefinidos:
  - Discretos:
    - `UITapGestureRecognizer`
    - `UISwipeGestureRecognizer`
  - Continuos:
    - `UIPinchGestureRecognizer`
    - `UIRotationGestureRecognizer`
    - `UIPanGestureRecognizer`
    - `UILongPressGestureRecognizer`
    - `UIScreenEdgePanGestureRecognizer`
- También podemos programar nuestros propios reconocedores de gestos.

UIGestureRecognizer

- **`UIGestureRecognizer`**: Clase base de la que derivan todos los reconocedores de gestos.
  - Es una clase abstracta: no crear instancias de esta clase, sino de sus subclases
- Creación:

  `initWithTarget:action:`
  - Cuando se reconozca el gesto, se ejecuta la acción dada del target indicado.
- Añadir nuevos target/action a ejecutar al reconocer el gesto, o quitarlos:

  `addTarget:action:`
  `removeTarget:action:`
- Información sobre el gesto:

  `locationInView:`
  `locationOfTouch:inView:`
  `numberOfTouches:`
  - Posición del gesto/toque en la view, número de toques del gesto.

- Propiedades:
  - `state`
  - `view`
  - `enabled`
  - `delegate`
  - . . .
- Dependencias entre gestos:

  `requireGestureRecognizeToFail:`
  - Cuando dos gestos empiezan igual hay que indicar cuál se desea reconocer antes.
    - Si falla el primero, reconozco el segundo.
      - Ejemplo: un swipe horizontal debe reconocerse sólo después de que la Z del zorro haya fallado.
- etc…

Target y Acción

- El mensaje de acción enviado a los targets cuando se reconoce un gesto puede:
  - no llevar parámetros
    - `(void)manejaGesto;`
  - o tomar como parámetro el objeto reconocedor:
    - `(void)manejaGesto:(UIGestureRecognizer*)recognizer;`

Añadir el Reconocedor a la UIView

- Supongamos que:
  - Ya hemos creado un objeto reconocedor para algún gesto.
  - Ya le hemos dicho a ese reconocedor que acciones tiene que invocar cuando reconozca ese gesto (usando `initWithTarget:action:` o `addTarget:action:`)

- Sólo falta decir cual es la `UIView` que debe vigilar el reconocedor.
  - **Programáticamente**:
    - Se hace con el método **`addGestureRecognizer:`** de `UIView`.
  - **Interface** Builder:
    - Enlazando la propiedad **`gestureRecognizers`** de la UIView con los objetos reconocedores de gestos:
      - Ctrl-Arrastrar desde una view hasta el objeto reconocedor.
  - A partir de este momento, el reconocedor analiza los eventos que ocurren en la `UIView`, y si detecta el gesto que le interesa, ejecuta la acciones programadas.

Desde Interface Builder

- Pueden añadirse reconocedores de gestos a las escenas de un storyboard o nib.
  - Arrastrándo los objetos reconocedores de gestos desde la librería de objetos hasta los ViewControllers.
- Con el Inspector de Atributos puede ajustarse las propiedades de los reconocedores de gestos.
- Pueden enlazarse los objetos reconocedores con los métodos IBAction existentes para que atiendan los gestos *(target-action)*.
  - O crear directamente nuevos métodos IBAction mediante Ctrl-Arrastrar desde un objeto reconocedor hasta el código de la clase en la que se desea crear el método IBAction.
- Se puede enlazar la propiedad **`gestureRecognizers`** de las UIView con los objetos reconocedores de gestos que la deben vigilar:
  - Ctrl-Arrastrar desde la view hasta el objeto reconocedor, y seleccionar la propiedad.

Ejemplo: Reconoce Tap

- Un ViewController crea un reconocedor de taps que al reconocer un tap llama a su método **`procesaTap`**.
- Y se lo añade a su top view.

```
- (void) procesaTap:(UITapGestureRecognizer *)sender {
    CGPoint pos = [sender locationInView:sender.view];
    NSLog(@"TAP en x=%i y=%i", (int)pos.x, (int)pos.y);
}

- (void)viewDidLoad {
    [super viewDidLoad];

    UITapGestureRecognizer *rec = [[UITapGestureRecognizer alloc]
                                   initWithTarget:self
                                   action:@selector(procesaTap:)];
    [self.view addGestureRecognizer:rec];
}
```

El reconocedor

¿Dónde pulsé en sender.view?

En este ejemplo sender.view es self.view

Reconocedor de taps

Ejecutar el método self.procesaTap:

Reconocer el gesto en la top view del VC

El Estado

- La propiedad **`state`** indica en que estado se encuentra el reconocedor.
  - El reconocedor está en el estado **`Possible`** hasta que empieza a reconocer un gesto.
  - Si el gesto que reconoce es **discreto**, pasa al estado **`Recognized`** cuando lo reconoce.
  - Si el gesto es **continuo**, pasa al estado **`Began`**, y después a **`Changed`**, hasta llegar finalmente a **`Ended`**.
    - pero esta secuencia puede terminar con **`Failed`** o **`Cancelled`**.

Gestos Continuos

- Los reconocedores de gestos continuos llaman a los manejadores (target/action) registrados cada vez que cambia el estado.
- Los manejadores deben comprobar el valor de **`state`** para decidir que deben hacer según el estado.

```
-(void) addLocation:(UILongPressGestureRecognizer*)sender {

    if (sender.state != UIGestureRecognizerStateBegan) return;

    // hacer cosas
}
```

Sólo se ejecuta cuando **comienza** el gesto Long Press

UITapGestureRecognizer

- Reconocer toques (golpe) rápidos.
- Propiedades para configurar el gesto a reconocer:

```
NSUInteger numberOfTapsRequired
NSUInteger numberOfTouchesRequired
```

UISwipeGestureRecognizer

- Reconoce un movimiento horizontal o vertical del dedo sobre la view.
- Propiedades para configurar el gesto a reconocer:

```
UISwipeGestureRecognizerDirection direction
NSUInteger numberOfTouchesRequired
```

UIPinchGestureRecognizer

- Reconoce pellizcos con dos dedos.
  - o la separación de los dos dedos.
- Propiedades con información sobre el gesto:

  `CGFloat` **`scale`**

  - es el valor de escala acumulado.
  - puede resetearse para borrar el valor acumulado.

  `CGFloat` **`velocity`** (read only)

UIRotationGestureRecognizer

- Reconoce giros con dos dedos.
- Propiedades:

  `CGFloat` **`rotation`**
  - es el valor de rotación acumulado.
  - puede resetearse para borrar el valor acumulado.

  `CGFloat` **`velocity`** (read only)

UIPanGestureRecognizer

- Reconoce movimientos (Drag) de uno o varios dedos.
- Propiedades para configurar el gesto a reconocer:

```
NSUInteger minimumNumberOfTouches
NSUInteger maximumNumberOfTouches
```

- Métodos informativos sobre el gesto:

```
-(CGPoint)translationInView:(UIView*)view
```

  - Distancia (valor acumulado) que se ha movido el dedo en la view dada.

```
-(void)setTranslation:(CGPoint)translation
               inView:(UIView*)view
```

  - Resetear o cambiar el valor acumulado.

```
-(CGPoint)velocityInView:(UIView*)view
```

UILongPressGestureRecognizer

- Reconoce pulsaciones largas
  - Es un gesto **continuo**.
    - por tanto, invoca los manejadores (target/action) si hay desplazamientos de los dedos tras la pulsación larga.
- Propiedades para configurar el gesto a reconocer:

```
CFTimeInterval minimumPressDuration
NSUInteger numberOfTapsRequired
NSInteger numberOfTouchesRequired
CGFloat allowableMovement
```

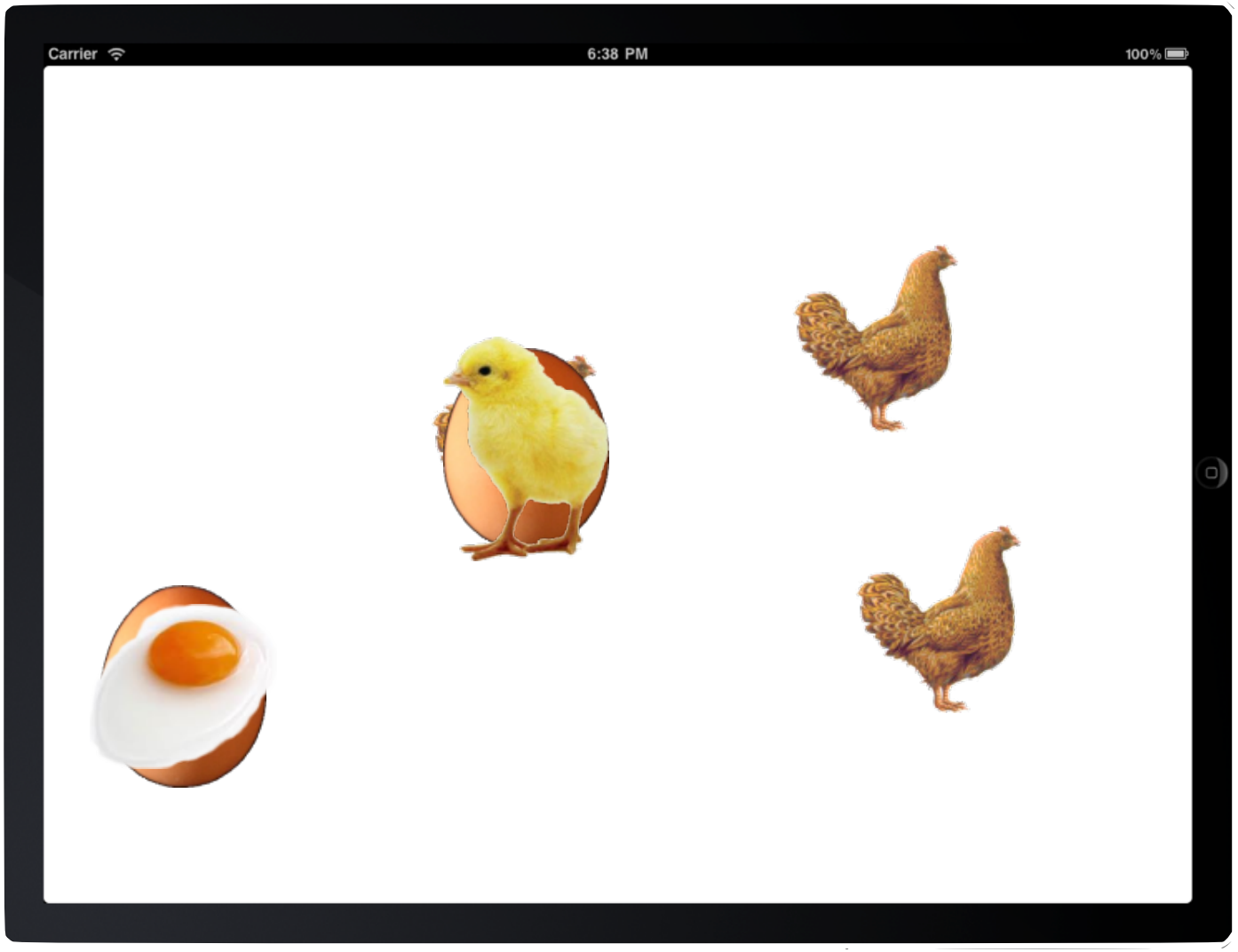
UIScreenEdgePanGestureRecognizer

- Reconoce un gesto Pan que comienza cerca del borde de la pantalla.
  - Algunas aplicaciones usan este gesto para realizar transiciones entre View Controllers.
- Propiedades para configurar el gesto a reconocer:
  `UIRectEdge edges`
  - El valor de **edges** es una máscara de bits usando los valores:
    - **UIRectEdgeNone**, **UIRectEdgeAll**, **UIRectEdgeTop**, **UIRectEdgeLeft**, **UIRectEdgeBottom** y **UIRectEdgeRight**.

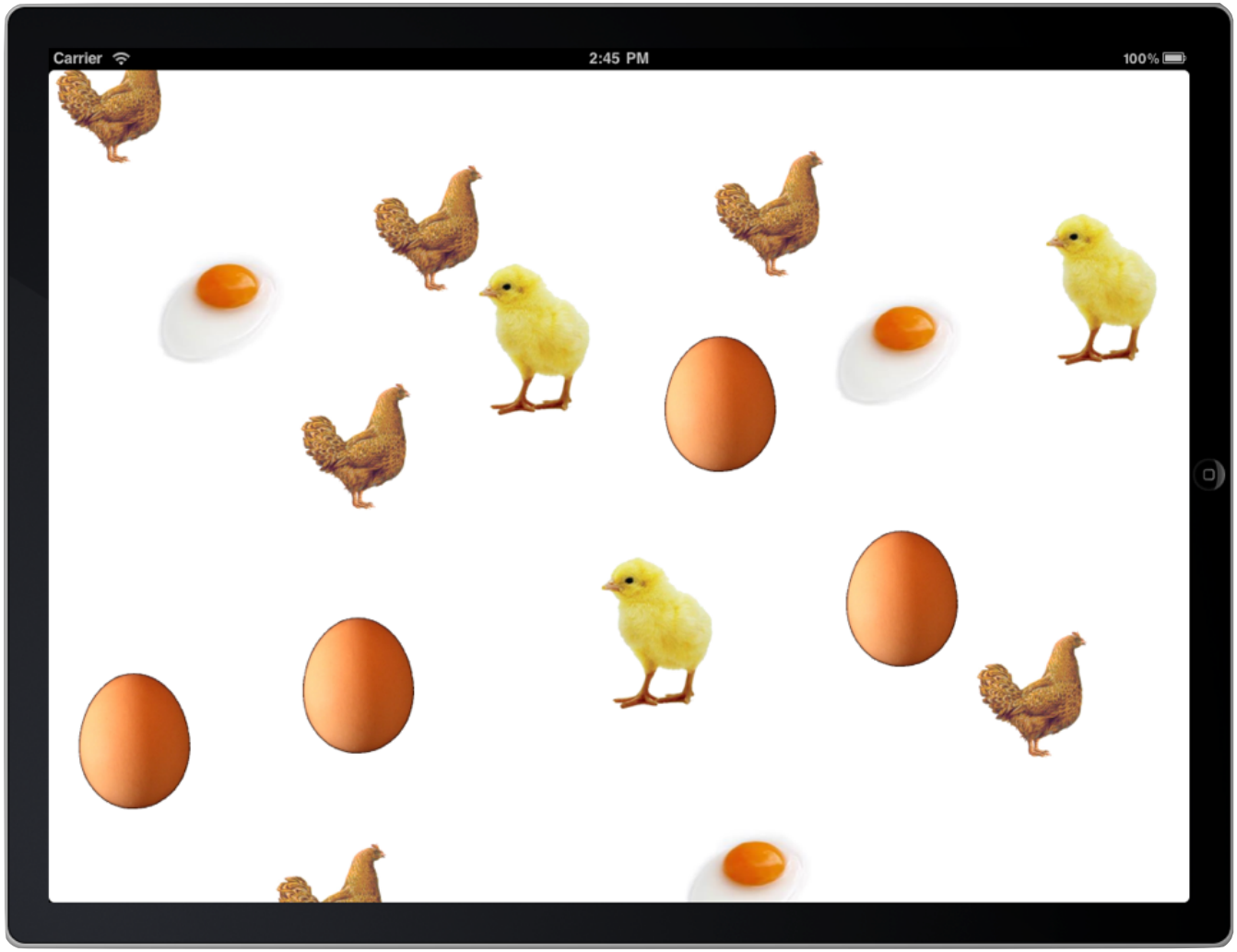
Demo

La Granja

Demo

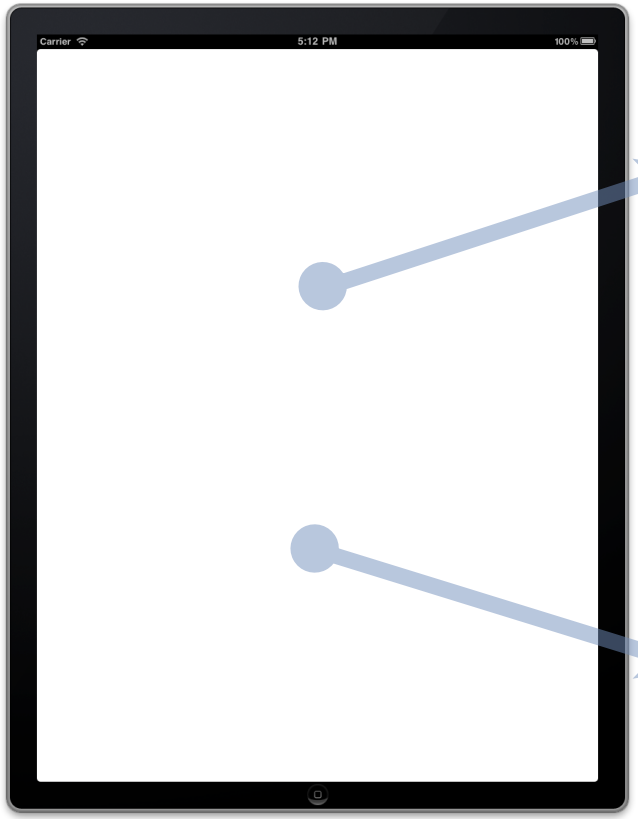


Demo



Reconocedores para el Fondo

- Los creo en **`viewDidLoad`** del ViewController.
  1. Reconocedor de un gesto **Tap** en el fondo
     - Llama a **`processRootTap`**
       - crea una **GHPView** (muestra una gallina)
  2. Reconocedor de un gesto **Swipe** en el fondo
     - Llama **`processRootSwipe`**
       - borra la pantalla
- (el fondo es la **`self.view`** del ViewController)



Reconocedor de Tap

processRootTap

Reconocedor de Swipe

processRootSwipe

```
- (void)viewDidLoad {
    [super viewDidLoad];


    // Crear reconocedor de Tap para crear gallina
    UITapGestureRecognizer *tapRec =
            [[UITapGestureRecognizer alloc]
                        initWithTarget:self
                                action:@selector(processRootTap:)];
    [tapRec setNumberOfTapsRequired:1]; // es el valor por defecto
    [self.view addGestureRecognizer:tapRec];


    // Crear reconocedor de Swipe para borrar todo
    UISwipeGestureRecognizer * swipeRec =
            [[UISwipeGestureRecognizer alloc]
                        initWithTarget:self
                                action:@selector(processRootSwipe:)];
    swipeRec.direction = UISwipeGestureRecognizerDirectionRight |
                         UISwipeGestureRecognizerDirectionLeft;
    [self.view addGestureRecognizer:swipeRec];
}
```

Manejador para crear gallina inicial

- **processRootTap** crea la **GHPView** que muestra una gallina, y la añade a **self.view** (el fondo):

```
- (void)processRootTap:(UITapGestureRecognizer *)sender {

    CGPoint pos = [sender locationInView:sender.view];

    CGRect rect = CGRectMake(pos.x, pos.y, 1, 1);
    GHPView *imgv = [[GHPView alloc] initWithFrame:rect];

    [self.view addSubview:imgv];
}
```

Borrar el Fondo

```
- (void) processRootSwipe:(UISwipeGestureRecognizer*)sender {

    [UIView transitionWithView:self.view
                      duration:0.5
                       options: UIViewAnimationOptionTransitionFlipFromLeft
                    animations:^{

                       for (UIView *subview in [self.view subviews]) {
                          [subview removeFromSuperview];
                       }

                    }
                    completion:nil];
}
```

La transición anima el cambio en la jerarquía de views

Quitamos todas las subviews de self.view

La Clase GHPView

- Es una clase propia que deriva de **UIImageView**
  - ampliada con una propiedad para indicar su estado:
    - SOY_UNA_GALLINA
    - SOY_UN_HUEVO
    - SOY_UN_POLLO
    - SOY_UN_HUEVO_FRITO

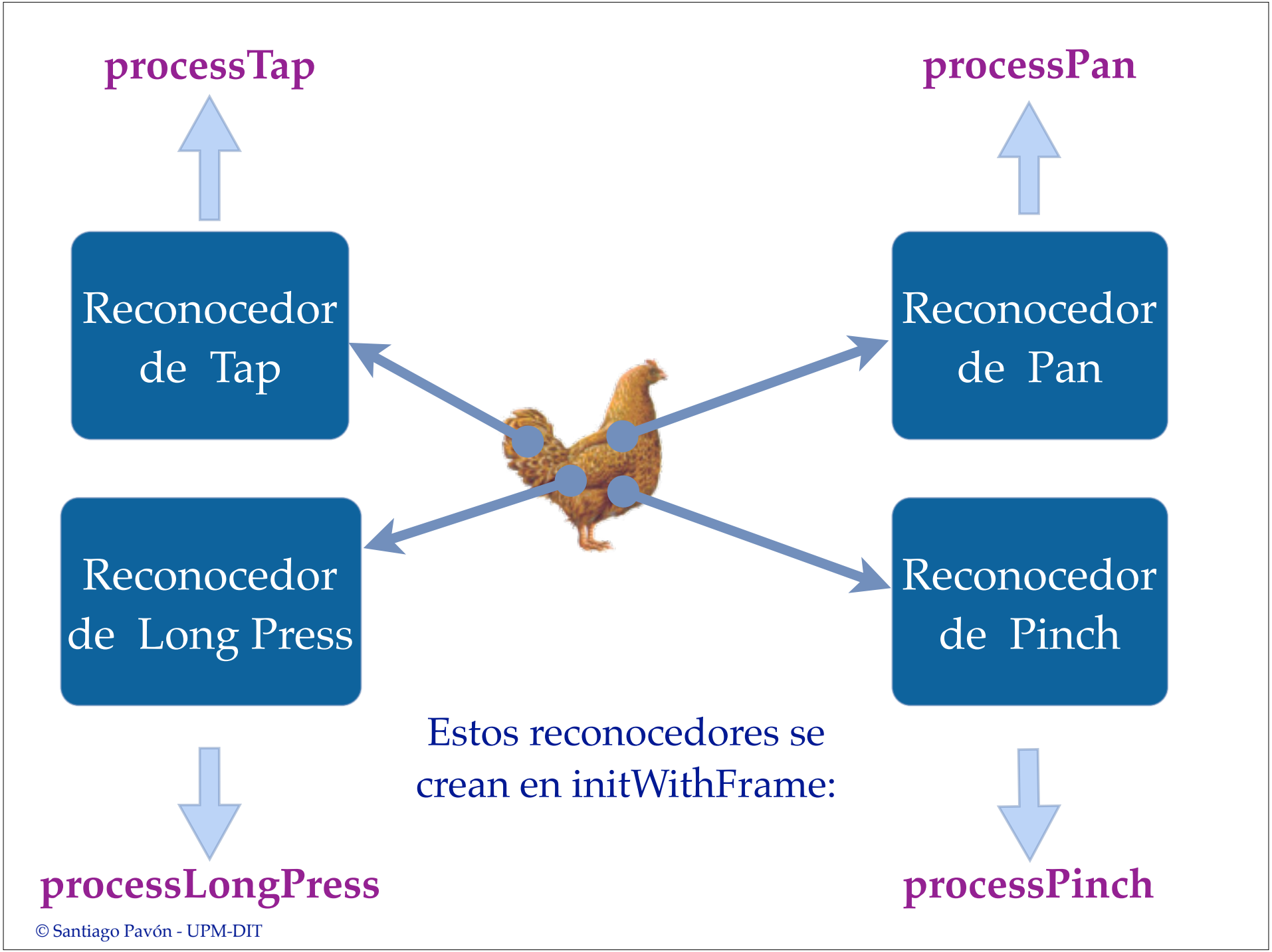
  - al cambiar el valor de la propiedad se cambia la imagen mostrada.

```
enum GHPState { GHP_I_AM_HEN, GHP_I_AM_EGG, GHP_I_AM_CHICKEN, GHP_I_AM_FRIED };

@interface GHPView ()
@property (nonatomic) enum GHPState ghpState;
@end

@implementation GHPView

- (instancetype)initWithFrame:(CGRect)frame {
    if (self = [super initWithFrame:frame]) {
        self.ghpState = GHP_I_AM_HEN; // Imagen inicial
    }
    return self;
}

- (void) setGhpState:(enum GHPState)ghpState { // Redefino el setter
    _ghpState = ghpState;
    switch (ghpState) {
      case GHP_I_AM_HEN:
           { self.image = [UIImage imageNamed:@"gallina.png"]; break; }
      case GHP_I_AM_EGG:
           { self.image = [UIImage imageNamed:@"huevo.png"];   break; }
      case GHP_I_AM_CHICKEN:
           { self.image = [UIImage imageNamed:@"pollo.png"];   break; }
      case GHP_I_AM_FRIED:
           { self.image = [UIImage imageNamed:@"frito.png"];   break; }
     }
   self.bounds = CGRectMake(0, 0, self.image.size.width, self.image.size.height);
}
@end
```

GHPView.m

Reconocedores de Gestos de GHPView

- Reconocedor de un gesto **Tap**
  - Llama a **processTap** para cambiar de gallina a huevo, y de huevo a pollo.
- Reconocedor de un gesto **Pan**
  - Llama a **processPan** para mover la imagen.
- Reconocedor de un gesto **Pinch**
  - Llama a **processPinch** para reescalar la imagen.
- Reconocedor de un gesto **Long Press**
  - Llama a **processLongPress** para cambiar de huevo a huevo frito.



Crear Reconocedores

```
// Necesito enterarme de los eventos.
[self setUserInteractionEnabled:YES];

// Crea reconocedor de Tap para ir de gallina a huevo, y de huevo a pollo
UITapGestureRecognizer *tapRec = [[UITapGestureRecognizer alloc]
                 initWithTarget:self action:@selector(processTap:)];
[self addGestureRecognizer:tapRec];

// Creo reconocedor de Long Press para freir huevo
UILongPressGestureRecognizer *lpRec = [[UILongPressGestureRecognizer alloc]
                 initWithTarget:self action:@selector(processLongPress:)];
[self addGestureRecognizer:lpRec];

// Crea reconocedor de Pan para mover imagen view
UIPanGestureRecognizer *panRec = [[UIPanGestureRecognizer alloc]
                 initWithTarget:self action:@selector(processPan:)];
[self addGestureRecognizer:panRec];

// Crea reconocedor de Pinch para escalar imagen view
UIPinchGestureRecognizer *pinchRec = [[UIPinchGestureRecognizer alloc]
                 initWithTarget:self action:@selector(processPinch:)];
[self addGestureRecognizer:pinchRec];
```

Por defecto UIImageView no atiende eventos

Propiedades privadas

```
@interface GHPView ()

@property (nonatomic) CGFloat scale;
@property (nonatomic) CGFloat translationX;
@property (nonatomic) CGFloat translationY;
 ....
@end


@implementation GHPView

- (id)initWithFrame:(CGRect)frame
{
    ....
    self.scale = 1;
    self.translationX = 0;
    self.translationY = 0;
    ....

}
....
@end
```

Valores acumulados de escala y translación por todos los gestos realizados anteriormente

Valores iniciales

ProcessTap

```
-(void)processTap:(UITapGestureRecognizer *)sender {

    switch (self.ghpState) {
      case GHP_I_AM_HEN:
         self.ghpState = GHP_I_AM_EGG;
         break;

      case GHP_I_AM_EGG:
         self.ghpState = GHP_I_AM_CHICKEN;
         break;

      case GHP_I_AM_CHICKEN:
         break;

      case GHP_I_AM_FRIED:
         break;
     }
}
```

Pasar de gallina a huevo

Pasar de huevo a pollo

ProcessLongPress

```
-(void)processLongPress:(UILongPressGestureRecognizer *)sender {

   if (sender.state != UIGestureRecognizerStateBegan) return;

   switch (self.ghpState) {
     case GHP_I_AM_HEN:
        break;

     case GHP_I_AM_EGG:
        self.ghpState = GHP_I_AM_FRIED;
        break;

     case GHP_I_AM_CHICKEN:
        break;
     case GHP_I_AM_FRIED:
        break;
     }
}
```

Long Press es un gesto continuo.
Solo me interesa el comienzo.

Freír el huevo

ProcessLongPress

```
-(void)processLongPress:(UILongPressGestureRecognizer *)sender {

    if (sender.state != UIGestureRecognizerStateBegan) return;

    switch (self.ghpState) {
        case GHP_I_AM_HEN:
            break;
        case GHP_I_AM_EGG: { // Llaves necesarias en ARC para indicar ambito
            [UIView animateWithDuration:0.5
                                  delay:0
                                options:UIViewAnimationOptionTransitionFlipFromLeft
                             animations:^{ self.ghpState = GHP_I_AM_FRIED; }
                             completion:nil];
            break;
        }
        case GHP_I_AM_CHICKEN:
            break;
        case GHP_I_AM_FRIED:
            break;
    }
}
```

(igual que la versión anterior pero con una animación)

ProcessPan

Translación realizada por el gesto sobre la vista padre para seguir los movimientos del dedo con precisión.

```
- (void)processPan:(UIPanGestureRecognizer *)sender
{
    CGPoint p = [sender translationInView:[sender.view superview]];

    self.translationX += p.x;
    self.translationY += p.y;

    [sender setTranslation:CGPointZero inView:sender.view];

    [self updateAffineTransforms];
}
```

Actualizar la propiedades privadas que me he creado

Resetear el valor en el reconocedor

Me he creado este método para aplicar todas las transformaciones según valores de las propiedades privadas

ProcessPinch

```
- (void)processPinch:(UIPinchGestureRecognizer *)sender
{
    CGFloat factor = sender.scale;

    self.scale *= factor;

    sender.scale = 1;

    [self updateAffineTransforms];
}
```

Factor de escala indicado por el gesto

Actualizar la propiedad privada

Resetear el valor en el reconocedor

Me he creado este método para aplicar todas las transformaciones según valores de las propiedades privadas

Aplicar transformaciones

```
- (void) updateAffineTransforms
{
    CGAffineTransform t1 =
        CGAffineTransformMakeTranslation(self.translationX,
                                         self.translationY);

    CGAffineTransform t2 =
        CGAffineTransformMakeScale(self.scale, self.scale);

    self.transform = CGAffineTransformConcat(t2,t1);
}
```

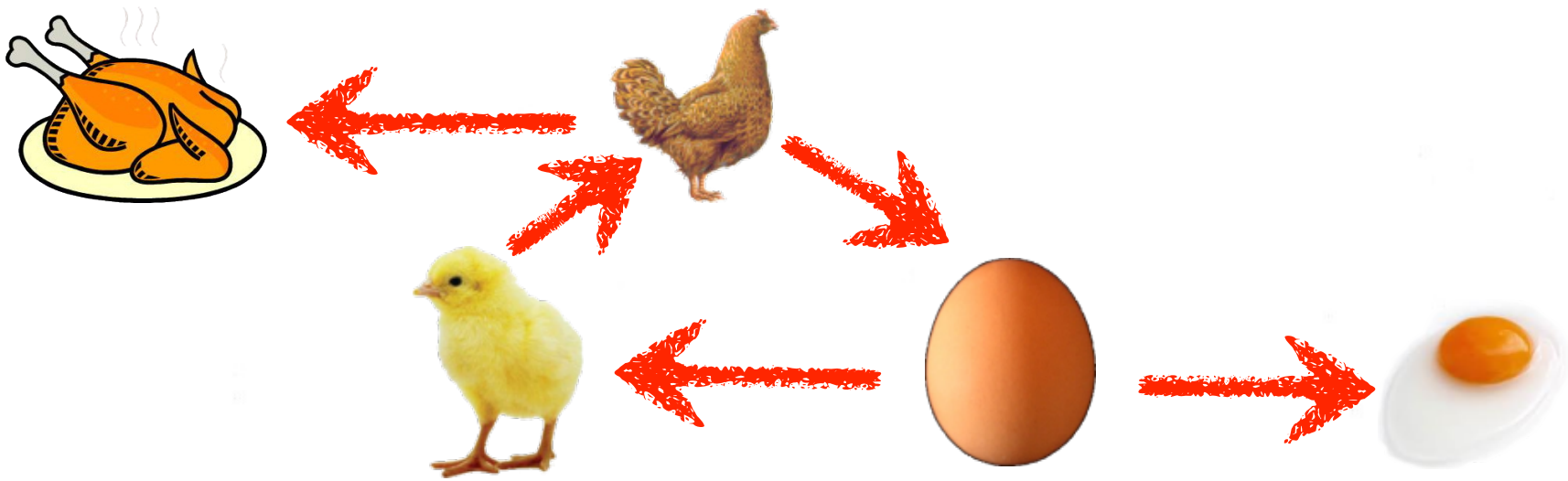
Valores guardados por processPan

Valores guardados por processPinch

Aplicar la concatenación de ambas transformaciones

Cambios en el Ejemplo

- El pollito puede crecer y convertirse en una gallina.
  - Y todo vuelva a empezar otra vez.
- El paso de huevo a huevo frito se hace de forma animada.
- Hay un reconocedor para rotar las imágenes.
  - El reescalado y la rotación no dan saltos bruscos.
- Las gallinas pueden asarse.



Información Adicional

Algunos métodos útiles

- Definidos en **UIView**
  - `-hitTest:withEvent:`
    - Devuelve la subview más profunda en la jerarquía en la que ocurrió el evento.
- Definidos en **NSObject**
  - `class`
  - `isKindOfClass`