ETSIT UPM

dit UPM

Desarrollo de Apps para iOS
`UIView` - Dibujar y Animaciones

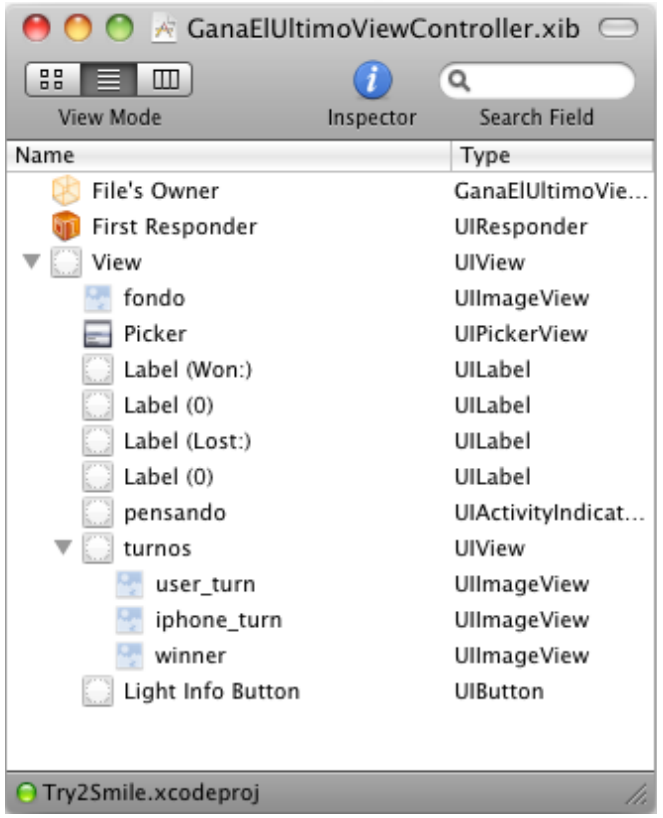
IWEB,LSWC 2013-2014
Santiago Pavón

ver: 2014.02.23

UIView

¿Qué es una UIView?

- Clase base de las views usadas para crear un GUI.
- El GUI es una jerarquía (árbol) de views.
- Propiedades para recorrer el árbol:
  - La view padre:
    `UIView *superview`
  - Lista (ordenada) de views hijas:
    `NSArray *subviews`



- La raíz de la jerarquía de views es un objeto **`UIWindow`**.

Construir el GUI

- La jerarquía de views puede construirse con el Interface Builder de Xcode.
- O programáticamente:

Inicializador designado de `UIView`. Usar con `alloc`.

```
- (id) initWithFrame:(CGRect)aRect;
```

Crear jerarquía de views.

```
- (void) addSubview:(UIView*)aView;
- (void) removeFromSuperview;

- (void) insertSubview:(UIView*)aView atIndex:(int)index;
- (void) insertSubview:(UIView*)aView belowSubview:(UIView*)otherView;
- (void) insertSubview:(UIView*)aView aboveSubview:(UIView*)otherView;
```

```
CGRect r = CGRectMake(50,50,120,40);

UILabel * l = [[UILabel alloc] initWithFrame:r];

l.text = @"Hola mundo";

[self.view addSubview:l];
```

El contenido de la View

- La view define un área rectangular en el que se puede dibujar.
- Si el contenido sobrepasa el rectángulo, se puede pintar o no (**`clipToBounds`**).
- Se puede controlar su nivel de transparencia (**`alpha`**).
- Puede ser opaca o no (**`opaque`**).
- La view entera se puede ocultar (**`hidden`**).
- Como reestablecer el contenido de una view cuando cambia su tamaño (**`contentMode`**):
  - reusar imagen cacheada de contenido actual reajustando tamaño y posición,
  - repintar otra vez el contenido.
- Transformaciones afines (**`transform`**).
- Tiene un color de fondo (**`backgroundColor`**).
- etc.
  - Ver la documentación

Eventos

- Podemos gestionar los eventos que ocurren en ella.
- Se puede configurar para:
  - ignorar los eventos del usuario.
  - decidir si soportar multi-touch.
  - establecer reconocedores de gestos.
  - bloquear el envío de eventos a otras views.
  - etc.

Gestión de Memoria

- Al añadir una view a la jerarquía se retiene.
- y al sacarla de la jerarquía se llama a release.
  - Cuidado que el contador de retenciones podría llegar a cero en este momento.
    - Podemos evitarlo llamando antes a retain (MRC) o asignado a una variable strong (ARC).

IBOutlet (solo MRC)

- Si una view es un IBOutlet:
  - La variable de instancia que la apunta la retiene

```
@property (retain) IBOutlet UILabel * l;
```

  - Para liberarla, llamamos a `release` y apuntamos a `nil`.

```
self.l = nil;
```

    - Ejemplo: las `UIViewController` sin memoria lo hacen en `viewDidUnload`.

  - Y sobreescribimos dealloc para llamar a `release`.

```
- (void) dealloc {
    [l release];
    [super dealloc];
}
```

Coordenadas

Algunas Funciones y Tipos **CG**

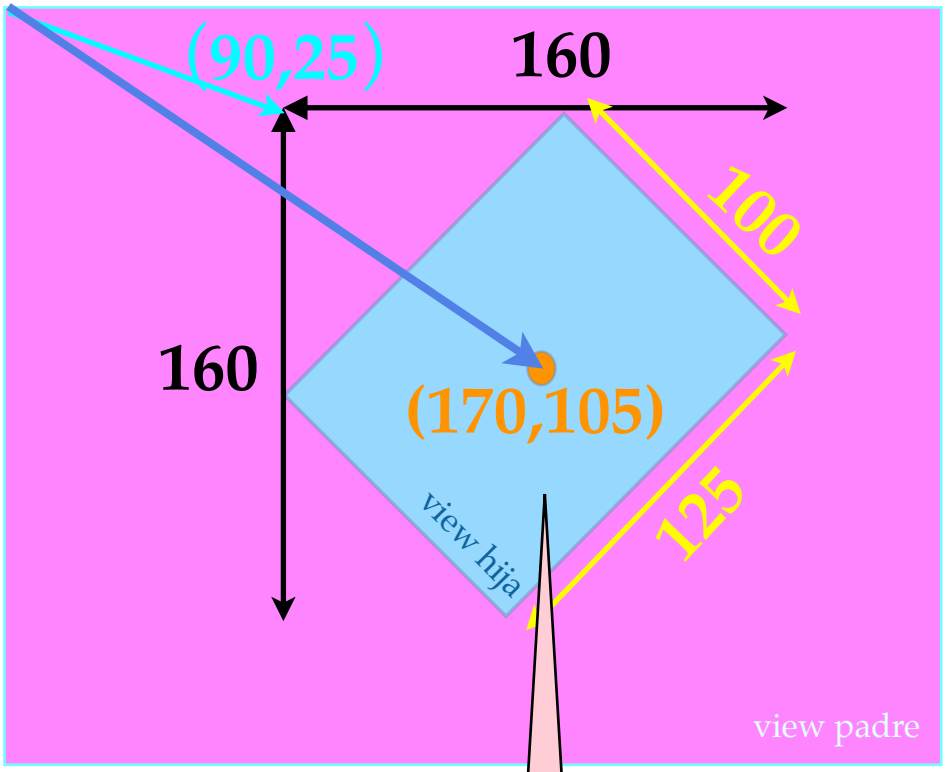
- **`CGFloat`**
  - Es un define a algún tipo de número real.
    - Usar siempre este tipo con gráficos. No usar: float, double, ...
- **`CGPoint`**
  - Es un struct de C con dos valores: `CGFloat x, CGFloat y`.
    - Representa un punto situado en (x,y).
- **`CGSize`**
  - Es un struct de C con dos valores: `CGFloat width, CGFloat height`.
    - Representa un tamaño.
- **`CGRect`**
  - Es un struct de C con los valores: `CGPoint origin, CGSize size`.
    - Representa un rectángulo con el origen y tamaño indicados.

- Métodos útiles para crear estas estructuras:

```
CGPoint p = CGPointMake(x,y);
CGSize  s = CGSizeMake(w,h);
CGRect  r = CGRectMake(x,y,w,h);
```

Coordenadas

- Origen de coordenadas situado arriba a la izquierda
- Las unidades son puntos, no pixeles.
  - Normalmente usaremos puntos cuando dibujemos.
  - Se ajustará el dibujo a la mayor resolución automáticamente.
    - la propiedad **`contentScaleFactor`** contiene número de pixels por punto de la `UIView`.
- Propiedades:
  - `CGRect` **`bounds`**
    - El origen y tamaño de la view en su sistema de coordenadas.
  - `CGPoint` **`center`**
    - El centro de la view en el sistema de coordenadas de la view padre.
  - `CGRect` **`frame`**
    - Rectángulo que contiene completamente a la view en el sistema de coordenadas de su view padre (la view hija puede estar girada).



View hija:

```
bounds = ((0,0),(100,125))
frame  = (90,25),(160,160))
center = (170,105)
```

`UIView`: Dibujar

Alternativas para Dibujar

- Podemos usar:
  - Quartz 2D
    - Pertenece al framework Core Graphics.
      - Consultar: *Quartz 2D Programming Guide*.
      - API en C.
    - El framework UIKit proporciona varios elementos para facilitar su uso.
      - `UIColor`, `UIfont`, `UIBezierPath`, ...
  - OpenGL ES

Dibujar con Quartz 2D

- Para hacer dibujos personales en la pantalla deberemos crear una subclase de **`UIView`**.
- Para dibujar hay que redefinir el método:

```
-(void)drawRect:(CGRect)rect;
```

  - Incluyendo en él las sentencias de pintado.
    - Se dibuja usando estructuras y funciones de C.
      - Pero UIKits nos proporciona algunos envoltorios útiles

```
CGColor c = [UIColor redColor].CGColor;
```

  - El rectángulo pasado como argumento indica la zona que hay que repintar.
- Nunca llamaremos directamente a **`drawRect`**
  - Cuando haya que repintar el contenido de la UIView, llamaremos a uno de estos métodos:

```
-(void)setNeedsDisplay;
-(void)setNeedsDisplayInRect:(CGRect)invalidRect;
```

    - Y el sistema llamará a `drawRect` en el siguiente ciclo de pintado.

Contexto Gráfico

- El pintado se hace usando un contexto gráfico.
- Independiza del elemento sobre el que se pinta
  - pantalla, imagen, impresora, PDF, …
- Se obtiene usando el método:

```
(CGContextRef)UIGraphicsGetCurrentContext();
```

  - Cada vez que se llama a `drawRect` se crea automáticamente un nuevo contexto gráfico.
    - Ese contexto sólo es válido para la actual llamada a `drawRect`.
    - No se puede guardar para usarlo más tarde.

Path

- Un path es un dibujo creado con líneas rectas y curvas, arcos, elipses y rectángulos.
  - Puede ser abierto o cerrado, relleno o no, con líneas sólidas o discontinuas, etc.
- Como se usa:
  - Crear un path.
  - Partimos de un punto inicial.
  - Añadimos tramos al path (líneas, arcos, ...).
    - Al final, podemos (opcional) cerrar el path.
  - Definir el color/tipo/patrón de las líneas, el color del relleno, zona de clip, etc...
  - Acabamos pintando el path creado.
    - Pintaremos las líneas que lo forman, el relleno o ambos.
- Un path también puede guardarse para reusarlo varias veces.

Dibujar una Copa

```
- (void)drawRect:(CGRect)rect {

    CGContextRef context = UIGraphicsGetCurrentContext();

    CGContextBeginPath(context);

    CGContextMoveToPoint(context, 10, 10);

    CGContextAddArc(context , 90, 10, 80, M_PI, M_PI_2, 1);
    CGContextAddLineToPoint(context, 95, 180);
    CGContextAddLineToPoint(context, 40, 200);
    CGContextAddLineToPoint(context, 170, 200);
    CGContextAddLineToPoint(context, 115, 180);
    CGContextAddLineToPoint(context, 120, 90);
    CGContextAddArc(context , 120, 10, 80, M_PI_2, 0, 1);

    CGContextClosePath(context);

    CGContextSetLineWidth(context, 3);

    CGContextSetStrokeColorWithColor(context,[UIColor magentaColor].CGColor);

    CGContextSetFillColorWithColor(context,[UIColor cyanColor].CGColor);

    CGContextDrawPath(context, kCGPathFillStroke);
}
```

Cambios de tamaño

- Si cambia el tamaño de una view:
  - Se redibuja su contenido si el valor de **`contentMode`** es **`UIViewContentModeRedraw`**.
  - Se reescala un bitmap cacheado que contiene la imagen actual si el valor de **`contentMode`** es **`UIViewContentModeScaleToFill`**, **`UIViewContentModeScaleAspectFit`**, **`UIViewContentModeTop`**.
  - Se dibuja su contenido ajustado a una posición si el valor de **`contentMode`** es **`UIViewContentModeCenter`**, **`UIViewContentModeLeft`**, **`UIViewContentModeRight`**, **`UIViewContentModeTop`**, **`UIViewContentModeRight`**, **`UIViewContentModeBottomLeft`**, **`UIViewContentModeBottomRight`**, **`UIViewContentModeTopLeft`**, **`UIViewContentModeTopRight`**.

UIKit Envoltorios

- UIKit proporcionan métodos útiles para simplificar el código de pintado que hay que programar:
  - ocultar la gestión de memoria, la configuración el contexto gráfico actual, ...

- Establecer el color de las líneas y de relleno:

```
[[UIColor blueColor] set];
```

- Establecer el color de las líneas:

```
[[UIColor greenColor] setStroke];
```

- Establecer el color de relleno:

```
[[UIColor redColor] setFill];
```

- Crear un color personalizado:

```
UIColor *c = [[UIColor alloc] initWithRed:0.2
                                    green:0.1
                                     blue:0.8
                                    alpha:1.0];
```

- Establecer el font:

```
UIFont *font1 = [UIFont systemFontOfSize:14];

UIFont *font2 = [UIFont fontWithName:@"Helvetica"
                                size:24.0];
```

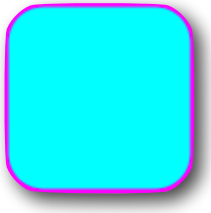
- Dibujar un rectángulo, relleno o solo el contorno:

```
UIRectFill(rect);
UIRectFrame(rect);
```

- Dibujar un path (rectángulo redondeado) en el contexto gráfico actual:

```
UIBezierPath *path =
      [UIBezierPath bezierPathWithRoundedRect:CGRectMake(10,10,100,100)
                                 cornerRadius:20];

path.lineWidth = 3;
[[UIColor magentaColor] setStroke];
[[UIColor cyanColor] setFill]
[path fill];
[path stroke];
```



- Dibujar un path (copa) en el contexto gráfico actual:

```
UIBezierPath *path = [UIBezierPath bezierPath];

[path moveToPoint:CGPointMake(10, 10)];
[path addArcWithCenter:CGPointMake(90, 10)
                radius:80
            startAngle:M_PI
              endAngle:M_PI_2
             clockwise:NO];
[path addLineToPoint:CGPointMake(95, 180)];
[path addLineToPoint:CGPointMake(40, 200)];
[path addLineToPoint:CGPointMake(170, 200)];
[path addLineToPoint:CGPointMake(115, 180)];
[path addLineToPoint:CGPointMake(120, 90)];
[path addArcWithCenter:CGPointMake(120, 10)
                radius:80
            startAngle:M_PI_2
              endAngle:0
             clockwise:NO];
[path closePath];
path.lineWidth = 3;
[[UIColor magentaColor] setStroke];
[[UIColor cyanColor] setFill];
[path fill];
[path stroke];
```

Dibujar Texto

```
NSString *text = @"hola";

CGPoint pos = CGPointMake(10,10);

UIFont *font = [UIFont systemFontOfSize:14];


[text drawAtPoint:pos withFont:font];
```

- Es mejor usar un objeto UILabel.

Dibujar Imágenes

```
UIImage *img = [UIImage imageNamed:@"fondo.png"];
```

Crear imagen usando fichero del bundle de la aplicación.

Pintar la imagen en la posición dada, usando el contexto gráfico actual.

```
[img drawAtPoint:punto];
```

```
[img drawInRect:rect];
```

Reescalar imagen y pintarla en el rectángulo dado.

```
[img drawAsPatternInRect:rect];
```

Pintar la imagen repetidas (tiled) veces para llenar el rectángulo dado.

- Es mejor usar un objeto `UIImageView`.

Crear UIImages

- Usando un fichero en el main bundle de la aplicación:

```
UIImage *img = [UIImage imageNamed:@"fondo.png"];
```

- Usando el path de un fichero:

```
NSArray *paths = NSSearchPathForDirectoriesInDomains(
                    NSDocumentDirectory,NSUserDomainMask,YES);
NSString *docs = [paths objectAtIndex:0];
NSString *filePath = [docs stringByAppendingPathComponent:@"mundo.jpg"];
UIImage *img = [[UIImage alloc] initWithContentOfFile:filePath];
```

- Desde un buffer de bytes:

```
NSData *data = [NSData dataWithContentsOfURL:url]
UIImage *img = [[UIImage alloc] initWithData:data];
```

- Dibujándola en un contexto gráfico:

```
UIGraphicsBeginImageContext(CGSizeMake(200,200));
// dibujar una copa usando funciones CGContext*
UIImage *img = UIGraphicsGetImageFromCurrentContext();
UIGraphicsEndImageContext();
```

Crear JPEG y PNG

- El contenido de una `UIImage` puede guardarse en un `NSData` en formato JPEG o PNG.

```
NSData* UIImageJPEGRepresentation(
      UIImage *image,
      CGFloat compressionQuality);


NSData* UIImagePNGRepresentation(
      UIImage *image);
```

Push y Pop del Contexto Gráfico

- Si se realizan modificaciones al estado del contexto gráfico, pero no se quiere perder el estado actual:
  - puede guardarse el estado actual del contexto gráfico con:

```
CGContextSaveGState(contexto);
```

  - realizar las modificaciones que se deseen en el contexto gráfico
  - y recuperar el estado inicial con:

```
CGContextRestoreGState();
```

- Ejemplo de uso:
  - `drawRect` invoca un método auxiliar que cambia el estado del contexto para hacer ciertos dibujos.
  - Este método hace un push al principio y un pop al final para que los cambios del estado sean locales a él.

```
- (void) drawAxis {

    CGContextRef context = UIGraphicsGetCurrentContext();

    // Guardar el estado del contexto actual
    CGContextSaveGState(context);

    // Realizo cambios en el estado del contexto
    CGContextSetLineWidth(context, 1);
    CGContextSetStrokeColorWithColor(context,[UIColor redColor].CGColor);

    . . .

    // Recupero el estado inicial
    CGContextRestoreGState();
}

. . .

- (void)drawRect:(CGRect)rect {
    [self drawAxis];
    [self drawObject];
}
```

Los cambios realizados por este método en el contexto gráfico no afectan a siguiente método

`UIView`: Animaciones y Transiciones

Animaciones

- UIView soporta animaciones al cambiar el valor de algunas propiedades:
  - **frame**, **bounds**, **center**, **transform**, **alpha**, **backgroundColor**, **contentStretch**

- La animación se define usando un método de clase y bloques.
  - El método de clase tiene parámetros para ajustar la animación:
    - retrasos, duración, curva de velocidad, ...
  - El bloque contiene el código que cambia el valor de las propiedades de la UIView.
  - Puede existir un ***Completion Block*** que se ejecuta al terminar la animación.

- Aunque la animación no haya terminado, el cambio en los valores de las propiedades es instantáneo.

```
+ (void)animateWithDuration:(NSTimeInterval)
                 animations:(void (^)(void))

+ (void)animateWithDuration:(NSTimeInterval)
                 animations:(void (^)(void))
                 completion:(void (^)(BOOL finished))

+ (void)animateWithDuration:(NSTimeInterval)
                      delay:(NSTimeInterval)
                    options:(UIViewAnimationOptions)
                 animations:(void (^)(void))
                 completion:(void (^)(BOOL finished))
```

Ejemplo

```
@property (nonatomic,weak) UILabel *label;


- (IBAction) anima {

    static int pos = 100;

    pos = 300-pos;

    CGPoint p = CGPointMake(pos, pos);

    [UIView animateWithDuration:2
            animations:^{label.center = p;}];
}
```

Transiciones

- También se pueden animar los cambios en la jerarquía de views, y los cambios de visibilidad.

```
+ (void)transitionFromView:(UIView *)
                    toView:(UIView *)
                  duration:(NSTimeInterval)
                   options:(UIViewAnimationOptions)
                completion:(void (^)(BOOL finished))
+ (void)transitionWithView:(UIView *)
                  duration:(NSTimeInterval)
                   options:(UIViewAnimationOptions)
                animations:(void (^)(void))
                completion:(void (^)(BOOL finished))
```

Opciones

`UIViewAnimationOptionLayoutSubviews`
Lay out subviews at commit time so that they are animated along with their parent.

`UIViewAnimationOptionAllowUserInteraction`
Allow the user to interact with views while they are being animated.

`UIViewAnimationOptionBeginFromCurrentState`
Start the animation from the current setting associated with an already in-flight animation.

`UIViewAnimationOptionRepeat`
Repeat the animation indefinitely.

`UIViewAnimationOptionAutoreverse`
Run the animation backwards and forwards.

`UIViewAnimationOptionOverrideInheritedDuration`
Force the animation to use the original duration value specified when the animation was submitted.

`UIViewAnimationOptionOverrideInheritedCurve`
Force the animation to use the original curve value specified when the animation was submitted.

`UIViewAnimationOptionAllowAnimatedContent`
Animate the views by changing the property values dynamically and redrawing the view.

`UIViewAnimationOptionShowHideTransitionViews`
This key causes views to be hidden or shown (instead of removed or added) when performing a transition.

`UIViewAnimationOptionCurveEaseInOut`
An ease-in ease-out curve causes the animation to begin slowly, accelerate and then slow again.

`UIViewAnimationOptionCurveEaseIn`
An ease-in curve causes the animation to begin slowly, and then speed up as it progresses.

`UIViewAnimationOptionCurveEaseOut`
An ease-out curve causes the animation to begin quickly, and then slow as it completes.

`UIViewAnimationOptionCurveLinear`
A linear animation curve causes an animation to occur evenly over its duration.

`UIViewAnimationOptionTransitionNone`
No transition is specified.

`UIViewAnimationOptionTransitionFlipFromLeft`
A transition that flips a view around its vertical axis from left to right.

`UIViewAnimationOptionTransitionFlipFromRight`
A transition that flips a view around its vertical axis from right to left.

`UIViewAnimationOptionTransitionCurlUp`
A transition that curls a view up from the bottom.

`UIViewAnimationOptionTransitionCurlDown`
A transition that curls a view down from the top.

`UIViewAnimationOptionTransitionCrossDissolve`
A transition that dissolves from one view to the next.

`UIViewAnimationOptionTransitionFlipFromTop`
A transition that flips a view around its horizontal axis from top to bottom.

`UIViewAnimationOptionTransitionFlipFromBottom`
A transition that flips a view around its horizontal axis from bottom to top.

`UIView.h`

Inicializacón de UIView

Inicialización

- Existen varios inicializadores de UIView:

  `-initWithFrame:`

  - Es el inicializador que llamamos (*típicamente*) explícitamente nosotros al crear un objeto UIView programáticamente.
  - Este inicializador los redefiniremos en nuestras clases derivadas de UIView para añadir las sentencias de configuración que necesitemos para las nuevas vistas.

  `-initWithCoder:`

  - Es el inicializador que se llama al reconstruir un objeto serializado.
    - Es decir, es el inicializador que se llama al cargar las views desde un fichero storyboard o nib.
  - **IMPORTANTE**: En estos casos **no** se invoca **`initWithFrame:`**
  - Las sentencias de configuración las podemos incluir en el método **`awakeFromNib`**.
    - Este método se invoca en todos los objetos cargados de un fichero NIB o storyboard.

- Patrón típico para no repetir el mismo código de configuración en el método **`initWithFrame`** y en el método **`awakeFromNib`**:

```
- (void) setup {
    // código de configuración del objeto UIView creado.
}


// Se llama al crear el objeto UIView programaticamante.
- (id)initWithFrame:(CGRect)aRect {
    self = [super initWithFrame:aRect];
    [self setup];
    return self;
}


// Se llama al crear el objeto UIView desde
// un fichero NIB o storyboard.
- (void)awakeFromNib {
    [self setup];
}
```

El método setup se ejecuta siempre, independientemente de como creemos el objeto UIView

Threads

Main Thread

- La manipulación de la interface de usuario debe hacerse sólo desde el main thread de la aplicación.
- Los métodos de UIView (y de otras clases de UIKit) sólo pueden invocarse desde el main thread.
- Únicamente la creación de los objetos UIView puede hacerse desde otro thread.