Desarrollo de Apps para iOS
Animaciones Y Transiciones

IWEB 2015-2016
Santiago Pavón

ver: 2015.09.23

`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 de UIView y closures.
  - 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 una closure ***Completion*** que se ejecuta al terminar la animación.

- Aunque la animación no haya terminado, el cambio en los valores de las propiedades es instantáneo.
  - Nota: si se modifican las restricciones de autolayout durante una animación, es necesario llamar al método **layoutIfNeeded** en el bloque de la animación para calcular inmediatamente los nuevos tamaños y posiciones.

```
class func animateWithDuration(_ duration: NSTimeInterval,
                     animations animations: () -> Void)


class func animateWithDuration(_ duration: NSTimeInterval,
                     animations animations: () -> Void,
                     completion completion: ((Bool) -> Void)?)


class func animateWithDuration(_ duration: NSTimeInterval,
                               delay delay: NSTimeInterval,
                         options options: UIViewAnimationOptions,
                     animations animations: () -> Void,
                     completion completion: ((Bool) -> Void)?)


class func animateWithDuration(_ duration: NSTimeInterval,
                               delay delay: NSTimeInterval,
         usingSpringWithDamping dampingRatio: CGFloat,
          initialSpringVelocity velocity: CGFloat,
                           options options: UIViewAnimationOptions,
                     animations animations: () -> Void,
                     completion completion: ((Bool) -> Void)?)
```

Ejemplo

```
@IBOutlet weak var label: UILabel!


var pos: CGFloat = 100

@IBAction func animate() {

    pos = 300 - pos

    let p = CGPointMake(pos,pos)

    UIView.animateWithDuration(1,
           delay: 0,
           usingSpringWithDamping: 0.6,
           initialSpringVelocity: 10,
           options: UIViewAnimationOptions.BeginFromCurrentState,
           animations: {self.label.center = p},
           completion: nil)
}
```

Transiciones

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

```
class func transitionFromView(_ fromView: UIView,
                        toView toView: UIView,
                      duration duration: NSTimeInterval,
                       options options: UIViewAnimationOptions,
                    completion completion: ((Bool) -> Void)?)


class func transitionWithView(_ view: UIView,
                      duration duration: NSTimeInterval,
                       options options: UIViewAnimationOptions,
                    animations animations: () -> Void,
                    completion completion: ((Bool) -> Void)?)
```

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`