Ejercicios de gráficos y animación en iOS

Índice

1. Generación de gráficas

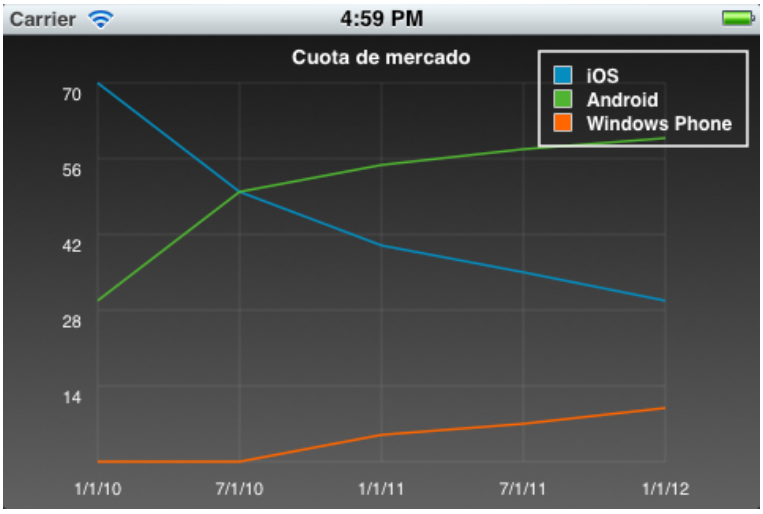
Vamos a crear una gráfica en iOS para mostrar la evolución de la cuota de mercado de diferentes plataformas móviles. Para ello utilizamos una sencilla librería: **S7Graph**, que consiste en una vista que mediante CoreGraphics genera la gráfica. Vamos a modificar dicha vista para mejorar el aspecto de la gráfica. Podemos ver la gráfica con su funcionamiento básico si abrimos la aplicación `Grafica` de las plantillas. Sobre esta aplicación se pide:

a) Vamos a dibujar la leyenda de la gráfica, para así saber a qué plataforma corresponde cada color. Para ello trabajaremos sobre el método `drawRect:` de `S7GraphView`, que es quien muestra el contenido de la gráfica. Empezaremos trabajando al final del método, donde mostraremos los elementos de la leyenda superpuestos sobre el contenido que ya se ha dibujado de la gráfica, en la esquina superior derecha. Buscar el comentario `TODO` referente al apartado *(a)*. Veremos que tenemos un bucle que itera por cada una de las líneas de la gráfica. Queremos mostrar para cada una de ellas un rectángulo con el color de la línea, y junto a él el texto indicando a qué plataforma corresponde dicha línea (iOS, Android y Windows Phone). Comenzaremos incluyendo en ese punto el código para dibujar un rectángulo con el color y dimensiones indicadas en los comentarios. El rectángulo deberá tener como color de relleno el color de la línea a la que hace referencia, y como borde color blanco.

b) En segundo lugar, vamos a dibujar el texto junto a cada rectángulo de la leyenda. Esto lo haremos en el lugar donde encontramos el comentario `TODO` referente al apartado *(b)*. El texto tendrá color de relleno blanco, y se dibujará en la posición indicada en los comentarios.

c) Para terminar de dibujar la leyenda, vamos a crear un marco que la englobe. Tras el bucle `for` sabremos cuál es la posición *y* final de la leyenda, y el texto que ha sido más largo (dentro del bucle se están comprobando las métricas del texto para tener esta información). Sabiendo esto, sabremos el tamaño que debe tener el rectángulo para que se adapte al número de líneas que tenemos, y a la leyenda de todas ellas. En el código encontramos un rectángulo llamado `recuadro` que nos da esa información. Vamos a dibujar un rectángulo blanco sin relleno con esas dimensiones.

d) Una vez dibujada la leyenda, vamos a terminar de decorar la gráfica dibujando un gradiente de fondo, en lugar de un color sólido. Esto deberemos hacerlo al principio de la función `drawRect:`, ya que el fondo debe dibujarse antes que el resto de elementos para que quede por debajo. Buscaremos el lugar donde tenemos un comentario de tipo `TODO` referente al apartado *(d)*, y dibujaremos ahí un gradiente desde gris oscuro hasta gris intermedio que varíe en la vertical, desde *y=0* hasta *y=320*.



Aspecto de la gráfica resultante

2. Animaciones (*)

En este ejercicio vamos a ver cómo hacer animaciones con CoreAnimation y directamente con vistas. Trabajaremos con el proyecto `Animaciones`. Se pide:

a) En la pantalla principal de la aplicación podemos ver una capa con la caráctula de la película "El Resplandor". Esta capa se inicializa en `viewDidLoad`. Configurar esa capa para decorarla tal como indica en los comentarios que encontramos en dicha función.

b) Vemos que hay cuatro botones, uno en cada esquina de la pantalla, con el texto *Ven aquí*. Al pulsar cualquiera de estos botones se ejecutará el método `botonVenAquiPulsado:`. Vamos a hacer que al pulsar cualquiera de estos botones, la capa se mueva mediante una animación a la posición central del botón pulsado. Incluye el código necesario en el método anterior.

c) La aplicación tiene también un botón *Carátula* que nos lleva a una vista modal que muestra en grande la caráctula de la película. Esta pantalla se implementa en el controlador `UACaratulaViewController`. En este controlador se definen dos vistas de tipo imagen: `vistaFrontal` y `vistaReverso`, con la imagen del anverso y el reverso de la carátula de la película. Sin embargo, en un primer momento sólo se muestra la imagen frontal. Vamos a hacer que al pulsar el botón *Girar* la carátula gire mediante una animación para cambiar entre anverso y reverso. Deberemos implementar esta transición con las facilidades de la clase `UIView` en el método `botonGirarPulsado:`.

Detalle de implementación

Podemos observar que en ese método distinguimos la vista que se está mostrando actualmente según si su propiedad `superview` es `nil` o no. Cuando una vista se muestra en pantalla siempre tiene una supervista. Si no tiene supervista quiere decir que no se está mostrando actualmente. Así podemos hacer esta distinción de forma sencilla.

d) Para terminar, vamos a implementar las funcionalidades de los botones *Zoom In* y *Zoom Out*. Con el primero de ellos veremos la portada ocupando toda la pantalla, mientras que con el segundo la reduciremos a la mitad de su tamaño. Esto deberemos hacerlo con las facilidades que nos proporciona la clase `UIView` para hacer animaciones, en los métodos `botonZoomInPulsado` y `botonZoomOutPulsado`. En el primero de ellos haremos con el tamaño de las vistas `vistaFrontal` y `vistaReverso` (propiedad `bounds`) sea *(0,0,320,416)*. En el segundo de ellos modificaremos estas propiedades a la mitad de tamaño: *(0,0,160, 208)*. Ten en cuenta que podemos modificar el tamaño de ambas vistas simultáneamente.