Tema 6

Posicionamiento y Mapas

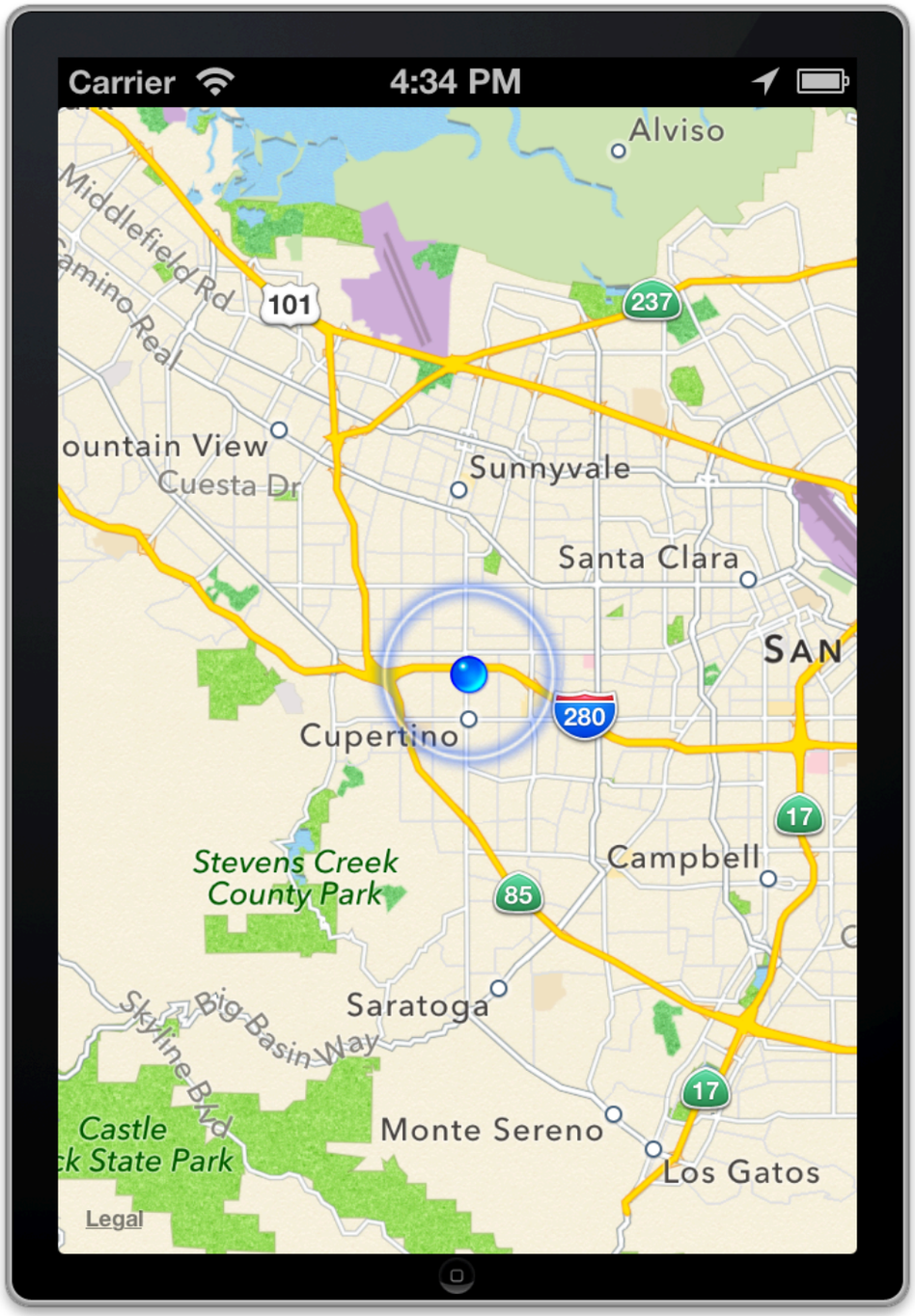
Location Framework

- El *framework* de posicionamiento nos proporciona la localización del usuario.

- Esta información es útil para mostrar recursos cercanos, orientar en rutas, realizar un seguimiento del desplazamiento, etc.

- Es posible acceder a la brújula para mejorar la experiencia en tareas de orientación.

Location y MapKit



Obteniendo la posición del usuario 1/2

- iOS 6 ofrece varios métodos de posicionamiento.
  - El posicionamiento en caso de cambio significativo provee de un modo de posicionamiento de bajo consumo de batería.
  - Los servicios de posicionamiento estándar ofrecen una forma más configurable de obtener la posición actual.

Obteniendo la posición del usuario 2/2

- La monitorización por región permite registrar los cambios entre los límites de un área definida.

- Es necesario incluir CoreLocation.framework en nuestra aplicación e incluir la cabecera
#import <CoreLocation/CoreLocation.h>

Requisitos

- Si queremos que sólo se utilice por dispositivos que puedan obtener la posición, es necesaria la modificación de Info.plist, añadiendo el campo UIRequiredDeviceCapabilities.

- Se debe incluir *location-services* si se desean los servicios de posicionamiento en general.

- Si se necesita exclusivamente el hardware GPS, la cadena a añadir será *gps*.

Iniciando el servicio de localización estándar

- Para utilizar el servicio de posicionamiento, es necesario inicializar un objeto de tipo CCLocationManager.

```
- (void)startStandardUpdates
{
    // Creamos un objeto de tipo Location Manager
    if (nil == locationManager)
        locationManager = [[CLLocationManager alloc] init];

    locationManager.delegate = self;
    locationManager.desiredAccuracy = kCLLocationAccuracyKilometer;



    // Fijamos un límite de 500 metros para la recepción de los nuevos eventos
    locationManager.distanceFilter = 500;
    [locationManager startUpdatingLocation];
}
```

desiredAccuracy y distanceFilter

- El parámetro **desiredAccuracy** (kCLLocationAccuracyBest) fija la precisión del posicionamiento.

- El parámetro **distanceFilter** kCIDistanceFilterNode) es la distancia mínima que el dispositivo se debe mover para generar un evento de actualización.

Iniciando el servicio de localización por cambio significativo

- Es similar a la inicialización normal, pero invocando el método **startMonitoringSignificantLocationChanges**

```
- (void)startSignificantChangeUpdates
{

    if (nil == locationManager)
          locationManager = [[CLLocationManager alloc] init];



    locationManager.delegate = self;
    [locationManager startMonitoringSignificantLocationChanges];
}
```

Recibiendo la información 1/2

- Cuando se produce una actualización de la posición, se avisa al método **locationManager:didUpdateLocation:fromLocation.**

```
// Método delegado para el protocolo CLLocationManagerDelegate.
- (void)locationManager:(CLLocationManager *)manager
didUpdateToLocation:(CLLocation *)newLocation fromLocation:(CLLocation *)oldLocation
{
    // Si es un evento reciente, desactivamos las actualizaciones para ahorrar batería
    NSDate* eventDate = newLocation.timestamp;
    NSTimeInterval howRecent = [eventDate timeIntervalSinceNow];



    if (abs(howRecent) < 15.0) {
          NSLog(@"latitude %+.6f, longitude %+.6f\n",
                      newLocation.coordinate.latitude,
                      newLocation.coordinate.longitude);
    }
}
```

Recibiendo la información 2/2

- Si existe algún error al recuperar la información, se informa al método locationManager:didFailWithError:.

- En ocasiones, el gestor de posicionamiento devuelve información cacheada, por lo que se debe comprobar el *timestamp* de todas las posiciones recibidas.

Consejos para reducir el consumo de batería

- Desactivar los servicios de posicionamiento cuando no se están usando.
- Usar el servicio de cambios significativos en lugar del servicio estándar cuando sea posible.
- Utilizar valores con baja precisión (distanceFilter).
- Desactivar los eventos de posicionamiento si la precisión no mejora durante un periodo de tiempo.

Información Geolocalizada

- La información de posicionamiento suele representarse como un par de valores, representando latitud y longitud del punto correspondiente del globo terráqueo.

- Esta información no es relevante para el usuario, que suele preferir información significativa, como la calle, la ciudad o el país del punto de interés.

Objetos Geocoder

- Se utiliza un servicio de Internet para convertir el par (latitud, longitud) en información significativa.
- En el iPhone convierte este par en un *placemark* o punto de interés.
- Es necesaria una conexión de datos para realizar la conversión.

Tipos

- Geocoder Inverso o ***Reverse*** (todas las versiones): Convierte un par latitud - longitud en un punto del mapa.

- Geocoding Directo o ***Forward*** (Disponible en iOS 5): Convierte nombres de lugares en pares de latitud - longitud.

Recomendaciones

- Es conveniente enviar como máximo una petición por cada acción de usuario.

- Si el usuario realizar acciones múltiples para el mismo punto, se debe reutilizar la información, en lugar de realizar varias peticiones.

- Si la posición se actualiza automáticamente, se debe enviar la petición sólo en el caso de que la diferencia en distancia o tiempo sea significativa.

- Si el usuario no visualiza los cambios inmediatamente, no es conveniente realizar peticiones de geocoding hasta que el usuario vuelva a estar activo.

Código Geocoder inverso

```
- (void)geocodeLocation:(CLLocation*)location forAnnotation:(MapLocation*)annotation
{
    if (!geocoder)
        geocoder = [[CLGeocoder alloc] init];

    [theGeocoder reverseGeocodeLocation:location completionHandler:
     ^(NSArray* placemarks, NSError* error){
         if ([placemarks count] > 0)
         {
             annotation.placemark = [placemarks objectAtIndex:0];

             // Add a More Info button to the annotation's view.
             MKPinAnnotationView*  view = (MKPinAnnotationView*)[map viewForAnnotation:annotation];
             if (view && (view.rightCalloutAccessoryView == nil))
             {
                 view.canShowCallout = YES;
                 view.rightCalloutAccessoryView = [UIButton buttonWithType:UIButtonTypeDetailDisclosure];
             }
         }
     }];
}
```

Geocoder Directo

- Las comas son recomendables pero no necesarias.
- Cuanta más información proporcionemos, menos ambiguo será el resultado.
- El manejador que se implemente debe estar preparado para recibir más de un resultado.

```
[geocoder geocodeAddressString:@"1 Infinite Loop"
     completionHandler:^(NSArray* placemarks, NSError* error){
         for (CLPlacemark* aPlacemark in placemarks)
         {
             // Process the placemark.
         }
     }];
```

MapKit

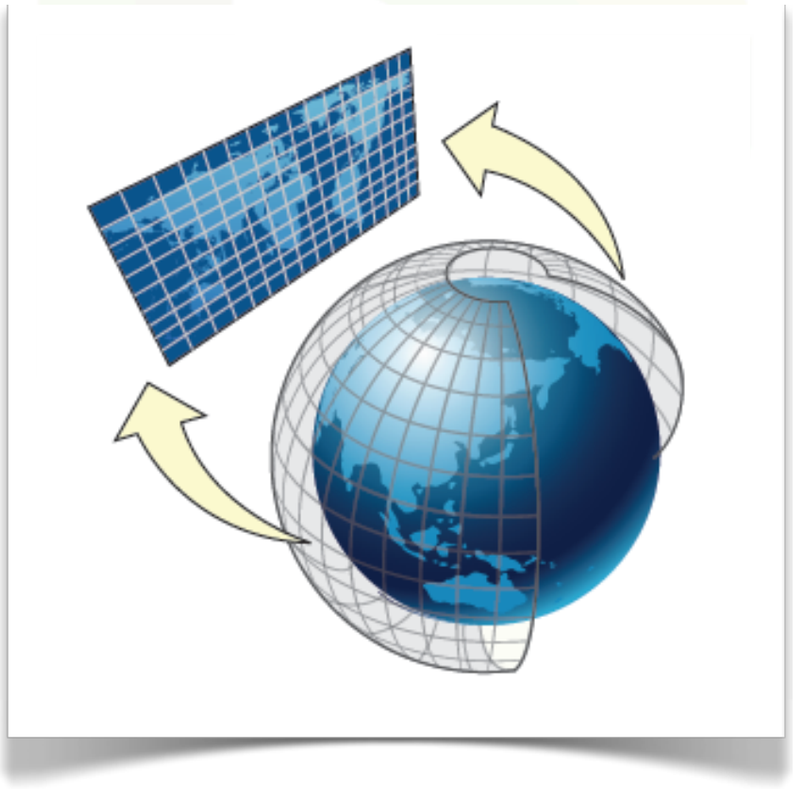
- Permite embeber un mapa interactivo en nuestra aplicación.

- Permite movernos por el mapa, hacer zoom en éste y realizar anotaciones.

- Es necesario añadir a nuestro proyecto el fichero **MapKit.framework** e incluir la línea **#import <MapKit/MapKit.h>** en el View Controller en el que vayamos a usar el mapa.

Principios geométricos 1/3

- **MapKit** utiliza la proyección **Mercator**, que es un tipo específico de proyección cilíndrica.



Principios geométricos 2/3

- Una de las ventajas de la proyección es que su escalado facilita la navegación por el mapa.

- Una coordenada del mapa es un par latitud/longitud correspondiente con la representación esférica de la Tierra. Las coordenadas se especifican con **CClocationCoordinate2D** y las áreas con **MKCoordinateSpan** y **MKCoordinateRegion**.

Principios geométricos 3/3

- Un punto del mapa es un par de valores (x, y) en la proyección **Mercator.** Se utilizan para simplificar los cálculos matemáticos entre puntos y áreas**.** Los puntos se especifican con **MKMapPoint** y las áreas con **MKMapSize** y **MKMapRect**.

- Un punto es una unidad gráfica asociada con el sistema de coordenadas de un objeto UIView. Por lo tanto, se puede asociar directamente la vista del mapa con la interfaz. Los puntos se especifican mediante **CGPoint** y las áreas mediante **MKMapSize** y **MKMapRet**.

Conversión del sistema de coordenadas

- Tabla con las diferentes representaciones de los datos y sus funciones.

Convert from	Convert to	Conversion routines
Map coordinates	Points	`convertCoordinate: toPointToView:` (MKMapView) `convertRegion: toRectToView:` (MKMapView)
Map coordinates	Map points	`MKMapPointForCoordinate`
Map points	Map coordinates	`MKCoordinateForMapPoint` `MKCoordinateRegionForMapRect`
Map points	Points	`pointForMapPoint:` (MKOverlayView) `rectForMapRect:` (MKOverlayView)
Points	Map coordinates	`convertPoint: toCoordinateFromView:` (MKMapView) `convertRect: toRegionFromView:` (MKMapView)
Points	Map points	`mapPointForPoint:` (MKOverlayView) `mapRectForRect:` (MKOverlayView)

El objeto MapView

- Se puede añadir un MapView mediante el Interface Builder.

- De manera programática, se debe crear una instancia de MKMapView e inicializarlo mediante el método initWithFrame: y añadirlo como una subvista dentro de la jerarquía de vistas.

Propiedades principales de MapView

- **region** (tipo MKCoordinateRegion). Define la parte visible del mapa. Es posible cambiar esta propiedad en cualquier momento, asignando a ésta un valor.

- **centerCoordinate**. Define la posición central del mapa.

Anotaciones

- Definir un objeto de anotación concreto.
  - Es posible utilizar **MKPointAnnotation** para crear una anotación simple. Contiene un *popup* con un título y un subtítulo.
  - Definir un objeto personalizado que siga el protocolo **MKAnnotation**.

Anotaciones 2

- Definir una vista de anotación para presentar la información en la pantalla.
    - Si la anotación se puede representar con una imagen estática, se debe crear una instancia de **MKAnnotationView** y asignar la imagen a la propiedad image.
    - Si se desea crear la anotación de chincheta, crear una instancia de **MKPinAnnotationView**.
    - Si la imagen estática es insuficiente, se puede crear una subclase de **MKAnnotationView** e implementar el código de dibujado para representarla.

Anotaciones 3

- Implementar el método **mapView:viewForAnnotation:** en el *delegate* del **MapView**.

- Añadir la anotación usando el método **addAnnotation:** o **addAnnotations:**.

Overlays

- Definir el Overlay apropiado (MKCircle, MKPolygon, MKPolyline o una subclase de MKShape o MKMultiPoint).

- Definir una vista para representar el overlay en la pantalla.

- Implementar el método mapView:viewForOverlay en el MapView delegate.

- Añadir el objeto al mapa mediante addOverlay:.

Rotando el mapa según la orientación

- iOS 5 permite rotar el mapa y seguir al usuario de una manera muy sencilla.

- Es suficiente con invocar al método: **setUserTrackingMode:MKUserTrackingModeFollowWithHeading animated:YES];**

- Permite el seguimiento del usuario con o sin *heading.*

Simulando cambios de posición

- El simulador ofrece varios tipos de simulación de posición. Destacamos las siguientes para probar nuestras aplicaciones:
  - **Custom Location.** Permite especificar un par de coordenadas.
  - **City Bicycle Ride.** Simula una vuelta en bici por la ciudad.
  - **City Run.** Simula un paseo a pie por la ciudad.
  - **Freeway Ride.** Simula una conducción en un vehículo de manera libre.