Stanford CS193p

Developing Applications for iOS
Fall 2011

Today

- Core Location
  Finding out where the device is
- MapKit
  Showing the location of things on a map
- Demo
  MapKit

Core Location

- Framework for managing location and heading
  No user-interface.
- Basic object is `CLLocation`
  `@properties: coordinate, altitude, horizontal/verticalAccuracy, timestamp, speed, course`
- Where (approximately) is this location?

```
@property (readonly) CLLocationCoordinate2D coordinate;
typedef {
    CLLocationDegrees latitude;    // a double
    CLLocationDegrees longitude;   // a double
} CLLocationCoordinate2D;

@property (readonly) CLLocationDistance altitude;  // meters
```

A negative value means "below sea level."

Core Location

- How close to that latitude/longitude is the actual location?

```
@property (readonly) CLLocationAccuracy horizontalAccuracy;   // in meters
@property (readonly) CLLocationAccuracy verticalAccuracy;     // in meters
```

  A negative value means the coordinate or altitude (respectively) is invalid.

```
kCLLocationAccuracyBestForNavigation;   // phone should be plugged in to power source
kCLLocationAccuracyBest;
kCLLocationAccuracyNearestTenMeters;
kCLLocationAccuracyHundredMeters;
kCLLocationAccuracyKilometer;
kCLLocationAccuracyThreeKilometers;
```

- The more accuracy you request, the more battery will be used

  Device "does its best" given a specified accuracy request
  Cellular tower triangulation (not very accurate, but low power)
  WiFi node database lookup (more accurate, more power)
  GPS (very accurate, lots of power)

Core Location

- Speed

```
@property (readonly) CLLocationSpeed speed;          // in meters/second
```

Note that the speed is instantaneous (not average speed).
Generally it's useful as "advisory information" when you are in a vehicle.
A negative value means "speed is invalid."

- Course

```
@property (readonly) CLLocationDirection course;    // in degrees, 0 is north, clockwise
```

Not all devices can deliver this information.
A negative value means "course is invalid."

- Time stamp

```
@property (readonly) NSDate *timestamp;
```

Pay attention to these since locations will be delivered on an inconsistent time basis.

- Distance between `CLLocations`

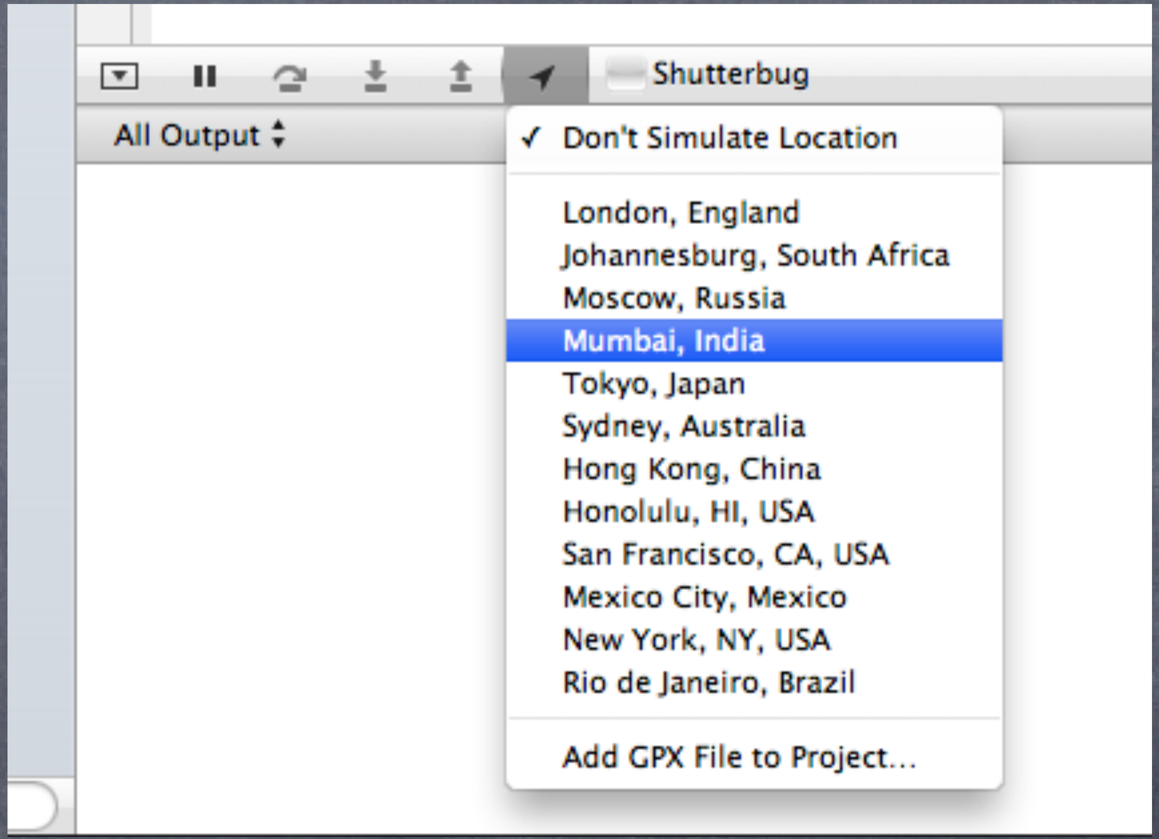
```
- (CLLocationDistance)distanceFromLocation:(CLLocation *)otherLocation; // in meters
```

Core Location

- How do you get a `CLLocation`?

  Almost always from a `CLLocationManager` (sent to you via its `delegate`).

  Can be tested in the simulator from Xcode.



Core Location

- How do you get a CLLocation?

  Almost always from a CLLocationManager (sent to you via its delegate).

  Can be tested in the simulator from Xcode.

- CLLocationManager

  General approach to using it:

  1. Check to see if the hardware you are on/user supports the kind of location updating you want.
  2. Create a CLLocationManager instance and set the delegate to receive updates.
  3. Configure the manager according to what kind of location updating you want.
  4. Start the manager monitoring for location changes.

Core Location

- Kinds of location monitoring

  Accuracy-based continual updates.

  Updates only when "significant" changes in location occur.

  Region-based updates.

  Heading monitoring.

Core Location

- Checking to see what your hardware can do

```
+ (BOOL)locationServicesEnabled;   // has the user enabled location monitoring in Settings?
+ (BOOL)headingAvailable;          // can this hardware provide heading info (compass)?
+ (BOOL)significantLocationChangeMonitoringAvailable;  // only if device has cellular?
+ (BOOL)regionMonitoringAvailable;   // only certain iOS4 devices
+ (BOOL)regionMonitoringEnabled;   // by the user in Settings
```

- Purpose

  When your application first tries to use location monitoring, user will be asked if it's okay to do so.
  You can provide a string which describes your app's purpose in using the location services.
  `@property (copy) NSString *purpose;`
  If the user denies you, the appropriate method above will return `NO`.

- Getting the information from the `CLLocationManager`

  You can just ask the `CLLocationManager` for the location or heading, but usually we don't.
  Instead, we let it update us when the location changes (enough) via its `delegate` ...

Core Location

- Accuracy-based continuous location monitoring

```
@property CLLocationAccuracy desiredAccuracy;  // always set this as low as possible

@property CLLocationDistance distanceFilter;
```

  Only changes in location of at least this distance will fire a location update to you.

- Starting and stopping the monitoring

```
- (void)startUpdatingLocation;
- (void)stopUpdatingLocation;
```

  Be sure to turn updating off when your application is not going to consume the changes!

- Get notified via the `CLLocationManager`'s `delegate`

```
- (void)locationManager:(CLLocationManager *)manager
     didUpdateToLocation:(CLLocation *)newLocation
            fromLocation:(CLLocation *)oldLocation;
```

Core Location

- Heading monitoring

  `@property CLLocationDegrees headingFilter;`

  Only changes in heading of at least this many degrees will fire a location update to you.

  `@property CLHeadingOrientation headingOrientation;`

  Heading of "zero degrees" is the heading of the "top" of the device.
  With this property, you can change that "top" (e.g. `CLDeviceOrientationLandscapeLeft`).

- Start the monitoring

  ```
  - (void)startUpdatingHeading;
  - (void)stopUpdatingHeading;
  ```

  Be sure to turn updating off when your application is not going to consume the changes!

- Get notified via the `CLLocationManager`'s `delegate`

  ```
  - (void)locationManager:(CLLocationManager *)manager
         didUpdateHeading:(CLHeading *)newHeading;
  ```

  What is `CLHeading`? ...

Core Location

- CLHeading

```
@property (readonly) CLLocationDirection magneticHeading;
@property (readonly) CLLocationDirection trueHeading;
```

Negative values mean "this heading is unreliable" (i.e. don't use it).
You won't get `trueHeading` if location services are turned off (e.g. by the user).

```
@property (readonly) CLLocationDirection headingAccuracy;   // in degrees
```

Basically how far off the magnetic heading might be from actual magnetic north.
A negative value means "this heading is not valid."

```
@property (readonly) NSDate *timestamp;
```

- Heading calibration user-interface

Automatically put up by iOS, but can be prevented by `CLLocationManager`'s delegate

```
- (BOOL)locationManagerShouldDisplayHeadingCalibration:(CLLocationManager *)manager;
```

Or dismissed (maybe after a timer or something) using `CLLocationManager` instance method

```
- (void)dismissHeadingCalibrationDisplay;
```

Core Location

- Error reporting to the delegate

```
- (void)locationManager:(CLLocationManager *)manager
        didFailWithError:(NSError *)error;
```

Not always a fatal thing, so pay attention to this delegate method.

```
kCLErrorLocationUnknown   // likely temporary, keep waiting (for a while at least)
kCLErrorDenied            // user refused to allow your application to receive updates
kCLErrorHeadingFailure    // too much local magnetic interference, keep waiting
```

Core Location

- Significant location change monitoring in `CLLocationManager`

  "Significant" is not strictly defined. Think vehicles, not walking. Likely uses cell towers.

  ```
  - (void)startMonitoringSignificantLocationChanges;
  - (void)stopMonitoringSignificantLocationChanges;
  ```

  Be sure to turn updating off when your application is not going to consume the changes!

- Get notified via the `CLLocationManager`'s `delegate`

  Same as for accuracy-based updating if your application is running.

- But this works even if your application is <u>not</u> running!

  (Or is in the background (we haven't talked about multitasking yet)).

  You will get launched and your application delegate will receive the message

  `application:didFinishLaunchingWithOptions:` with an options dictionary that will contain

  `UIApplicationLaunchOptionsLocationKey`

  Create a `CLLocationManager` (if you don't have one), then get the latest location via

  ```
  @property (readonly) CLLocation *location;
  ```

  If you are running in the background, don't take too long (a few seconds)!

Core Location

- Region-based location monitoring in CLLocationManager

```
- (void)startMonitoringForRegion:(CLRegion *) desiredAccuracy:(CLLocationAccuracy);
- (void)stopMonitoringForRegion:(CLRegion *);
```

- Get notified via the CLLocationManager's delegate

```
- (void)locationManager:(CLLocationManager *)manager didEnterRegion:(CLRegion *)region;
- (void)locationManager:(CLLocationManager *)manager didExitRegion:(CLRegion *)region;
   - (void)locationManager:(CLLocationManager *)manager
monitoringDidFailForRegion:(CLRegion *)region
                 withError:(NSError *)error;
```

- Works even if your application is not running!

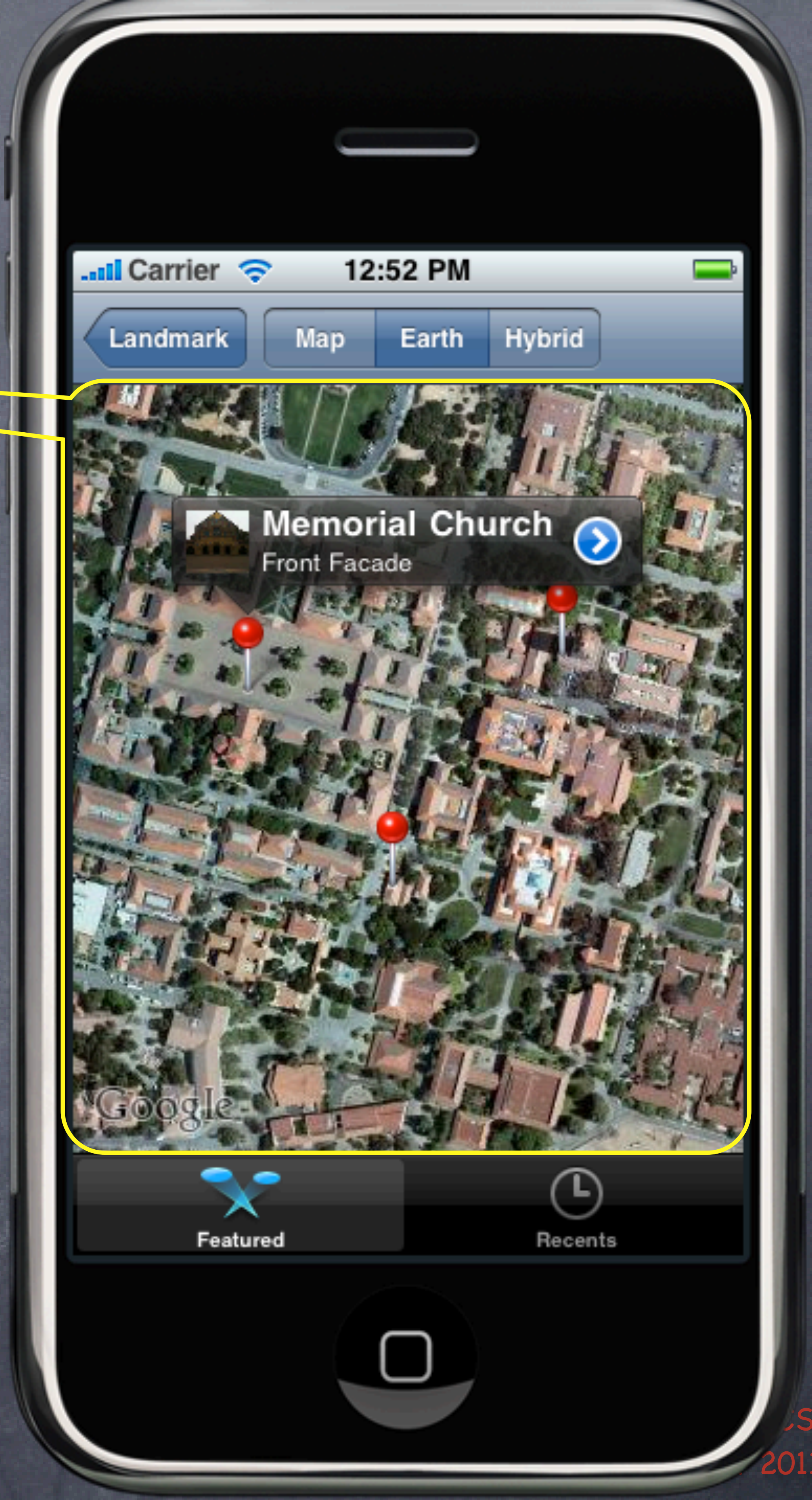
  In exactly the same way as "significant location change" monitoring.
  The set of monitored regions persists across application termination/launch.
  `@property (readonly) NSSet *monitoredRegions;` // property on CLLocationManager

Core Location

- CLRegions are tracked by name
  Because they survive application termination/relaunch.
- Regions (currently) require large location changes to fire
  Probably based on same technology as "significant location change" monitoring.
  Likely both of these "fire" when a new cell tower is detected.
  Definitely they would not use GPS (that would be very expensive power-wise).
- Region monitoring size limit
  `@property (readonly) CLLocationDistance maximumRegionMonitoringDistance;`
  Attempting to monitor a region larger than this (radius in meters) will generate an error (which will be sent via the `delegate` method mentioned on previous slide).
  If this property returns a negative value, then region monitoring is not working.

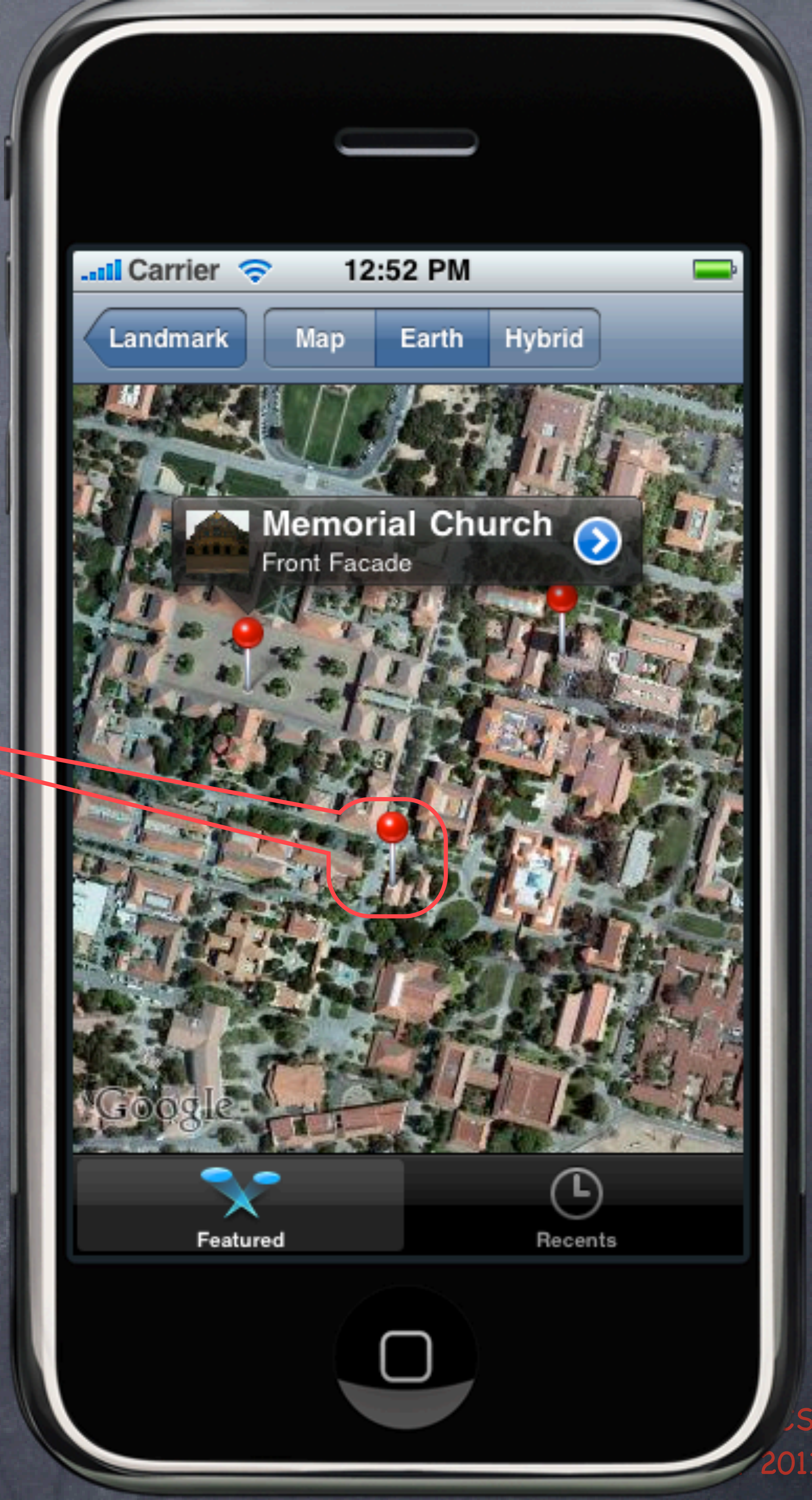
Map Kit

- MKMapView displays a map



Map Kit

- MKMapView displays a map
- The map can have annotations on it
  Each annotation is simply a coordinate, a title and a subtitle.
  They are displayed using an MKAnnotationView
  (MKPinAnnotationView shown here).

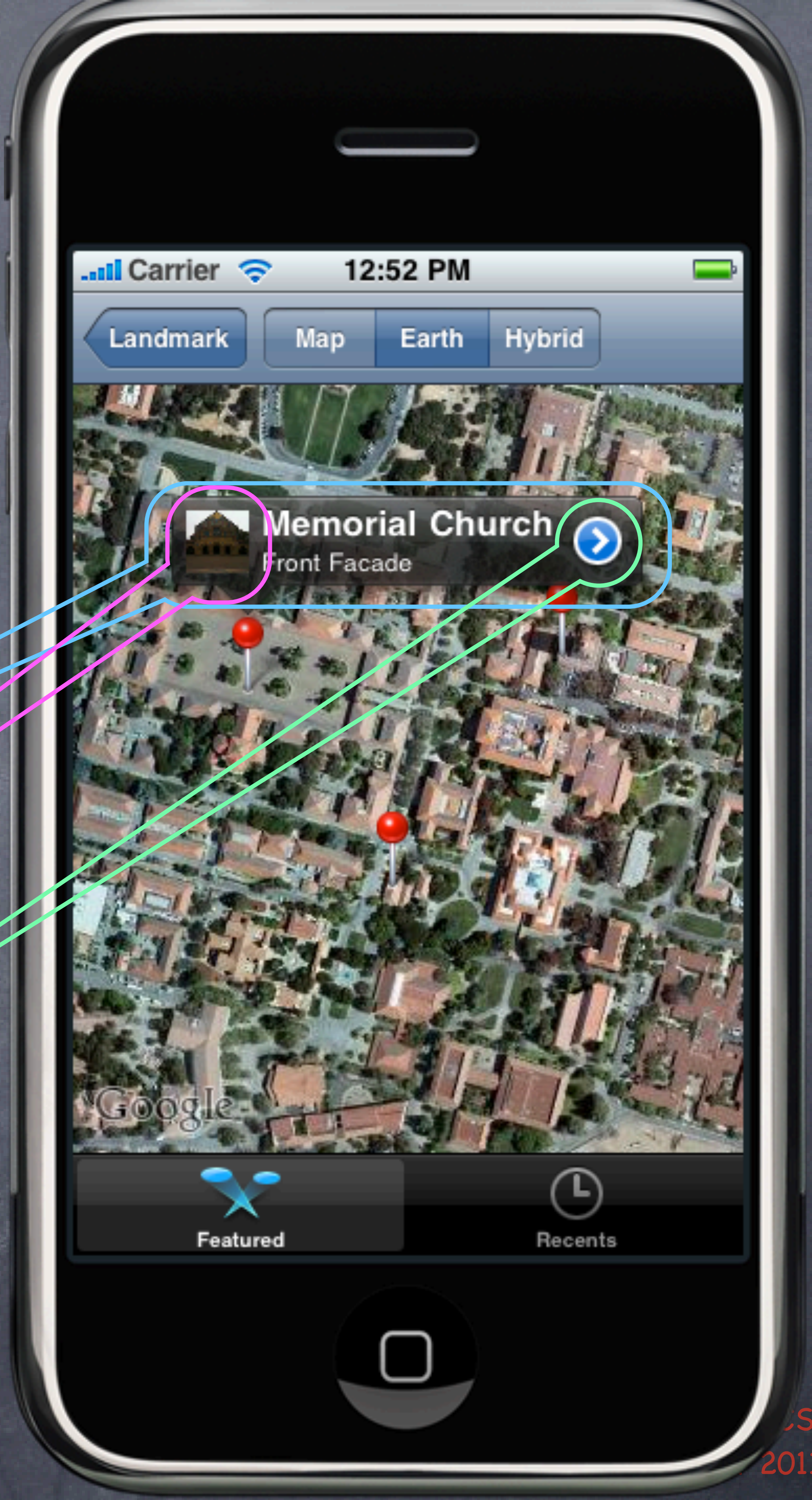


Map Kit

- MKMapView displays a map

- The map can have annotations on it
  Each annotation is simply a coordinate, a title and a subtitle.
  They are displayed using an MKAnnotationView
  (MKPinAnnotationView shown here).

- Annotations can have a callout
  It appears when the annotation view is clicked.
  By default just shows the title and subtitle.
  But you can add left and right accessory views.
  In this example, left is a UIImageView,
  right is a UIButton (UIButtonTypeDetailDisclosure)



MKMapView

- Create with alloc/init or drag from Library in Xcode
- Displays an array of objects which implement MKAnnotation

```
@property (readonly) NSArray *annotations; // contains id <MKAnnotation> objects
```

- MKAnnotation protocol

```
@protocol MKAnnotation <NSObject>
@property (readonly) CLLocationCoordinate2D coordinate;
@optional
@property (readonly) NSString *title;
@property (readonly) NSString *subtitle;
@end

typedef {
    CLLocationDegrees latitude;
    CLLocationDegrees longitude;
} CLLocationCoordinate2D;
```

MKAnnotation

- Note that the annotations property is readonly

  `@property (readonly) NSArray *annotations;` // contains id <MKAnnotation> objects

  Must add/remove annotations explicitly

```
- (void)addAnnotation:(id <MKAnnotation>)annotation;
- (void)addAnnotations:(NSArray *)annotations;
- (void)removeAnnotation:(id <MKAnnotation>)annotation;
- (void)removeAnnotations:(NSArray *)annotations;
```

- Generally a good idea to add all your annotations up-front

  Allows the MKMapView to be efficient about how it displays them

  Annotations are light-weight, but annotation views are not.

  Luckily MKMapView reuses annotation views similar to how UITableView reuses cells.

MKAnnotation

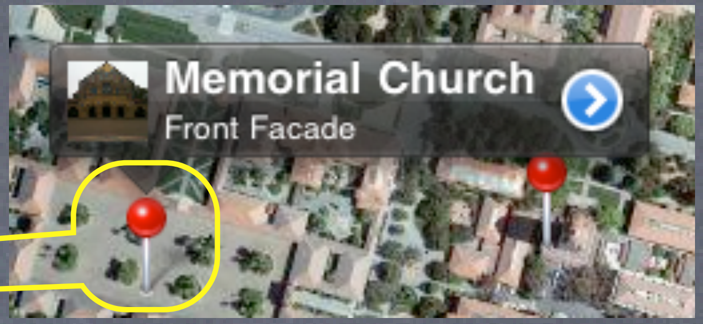
- What do annotations look like on the map?

  By default they look like a pin.

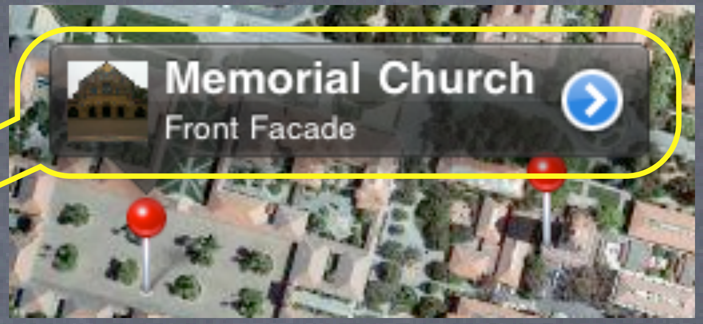
  Annotations are drawn using an MKAnnotationView subclass.

  The default one is MKPinAnnotationView (which is why they look like pins).

  You can create your own or set properties on existing MKAnnotationViews to modify the look.



MKAnnotation



- What do annotations look like on the map?

  By default they look like a pin.

  Annotations are drawn using an MKAnnotationView subclass.

  The default one is MKPinAnnotationView (which is why they look like pins).

  You can create your own or set properties on existing MKAnnotationViews to modify the look.

- What happens when you touch on an annotation (e.g. the pin)?

  Depends on the MKAnnotationView that is associated with the annotation (more on this later).

  By default, nothing happens, but if canShowCallout is YES in the MKAnnotationView, then
  a little box will appear showing the annotation's title and subtitle.

  And this little box (the callout) can be enhanced with left/rightCalloutAccessoryViews.

  The following delegate method is also called…

  ```
  - (void)mapView:(MKMapView *)sender didSelectAnnotationView:(MKAnnotationView *)aView;
  ```

  This is a great place to set up the MKAnnotationView's callout accessory views lazily.

  For example, you might want to wait until this method is called to download an image to show.

MKAnnotationView

- How are MKAnnotationViews created & associated w/annotations?

Very similar to UITableViewCells in a UITableView.
Implement the following MKMapViewDelegate method (if not implemented, returns a pin view).

```
- (MKAnnotationView *)mapView:(MKMapView *)sender
            viewForAnnotation:(id <MKAnnotation>)annotation
{
    MKAnnotationView *aView = [sender dequeueReusableAnnotationViewWithIdentifier:IDENT];
    if (!aView) {
        aView = [[MKPinAnnotationView alloc] initWithAnnotation:annotation
                                                 reuseIdentifier:IDENT];
        // set canShowCallout to YES and build aView's callout accessory views here
    }
    aView.annotation = annotation; // yes, this happens twice if no dequeue
    // maybe load up accessory views here (if not too expensive)?
    // or reset them and wait until mapView:didSelectAnnotationView: to load actual data
    return aView;
}
```

You can see why you might want to only show visible annotations (to keep view count low)

MKAnnotationView

- MKAnnotationView

  Interesting properties (all nonatomic, strong if a pointer):

  ```
  @property id <MKAnnotation> annotation;  // the annotation; treat as if readonly
  @property UIImage *image;  // instead of the pin, for example
  @property UIView *leftCalloutAccessoryView;  // maybe a UIImageView
  @property UIView *rightCalloutAccessoryView;  // maybe a "disclosure" UIButton
  @property BOOL enabled;  // NO means it ignores touch events, no delegate method, no callout
  @property CGPoint centerOffset;  // where the "head of the pin" is relative to the image
  @property BOOL draggable;  // only works if the annotation implements setCoordinate:
  ```

- If you set one of the callout accessory views to a UIControl

  e.g. `aView.rightCalloutAccessoryView = [UIButton buttonWithType:UIButtonTypeDetailDisclosure];`

  The following MKMapViewDelegate method will get called when the accessory view is touched ...

  ```
                - (void)mapView:(MKMapView *)sender
                 annotationView:(MKAnnotationView *)aView
  calloutAccessoryControlTapped:(UIControl *)control;
  ```

MKAnnotationView

- Using didSelectAnnotationView: to load up callout accessories

  Example ... downloaded thumbnail image in leftCalloutAccessoryView.
  Create the UIImageView and assign it to leftCalloutAccessoryView in mapView:viewForAnnotation:.
  Reset the UIImageView's image to nil there as well.

  Then load the image on demand in mapView:didSelectAnnotationView: ...

```
- (void)mapView:(MKMapView *)sender didSelectAnnotationView:(MKAnnotationView *)aView
{
    if ([aView.leftCalloutAccessoryView isKindOfClass:[UIImageView class]]) {
        UIImageView *imageView = (UIImageView *)aView.leftCalloutAccessoryView;
        imageView.image = ...;   // if you do this in a GCD queue, be careful, views are reused!
    }
}
```

MKMapView

- Configuring the map view's display type

```
@property MKMapType mapType;
MKMapTypeStandard, MKMapTypeSatellite, MKMapTypeHybrid;
```

- Showing the user's current location

```
@property BOOL showsUserLocation;
@property (readonly) BOOL isUserLocationVisible;
@property (readonly) MKUserLocation *userLocation;
```

`MKUserLocation` is an object which conforms to `MKAnnotation` which holds the user's location.

- Restricting the user's interaction with the map

```
@property BOOL zoomEnabled;
@property BOOL scrollEnabled;
```

MKMapView

- Controlling the region the map is displaying

```
@property MKCoordinateRegion region;
typedef struct {
    CLLocationCoordinate2D center;
    MKCoordinateSpan span;
} MKCoordinateRegion;
typedef struct {
    CLLocationDegrees latitudeDelta;
    CLLocationDegrees longitudeDelta;
}
- (void)setRegion:(MKCoordinateRegion)region animated:(BOOL)animated; // animate
```

- Can also set the center point only

```
@property CLLocationCoordinate2D centerCoordinate;
- (void)setCenterCoordinate:(CLLocationCoordinate2D)center animated:(BOOL)animated;
```

MKMapView

- Map loading notifications

  Remember that the maps are downloaded from Google earth.

  ```
  - (void)mapViewWillStartLoadingMap:(MKMapView *)sender;
  - (void)mapViewDidFinishLoadingMap:(MKMapView *)sender;
  - (void)mapViewDidFailLoadingMap:(MKMapView *)sender withError:(NSError *)error;
  ```

- Lots of C functions to convert points, regions, rects, etc.

  See documentation, e.g. `MKMapRectContainsPoint`, `MKMapPointForCoordinate`, etc.

Overlays

- Overlays

  Mechanism is similar to annotations (uses MKOverlayView instead of MKAnnotationView).

  ```
  - (void)addOverlay:(id <MKOverlay>)overlay;       // also addOverlays:(NSArray *)
  - (void)removeOverlay:(id <MKOverlay>)overlay;   // also removeOverlays:(NSArray *)
  ```

- MKOverlay protocol

  Protocol which includes MKAnnotation plus …

  ```
  @property (readonly) MKMapRect boundingMapRect;
  - (BOOL)intersectsMapRect:(MKMapRect)mapRect; // optional, uses boundingMapRect otherwise
  ```

- Overlays are associated with MKOverlayViews via delegate

  Just like annotations are associated with MKAnnotationViews …

  ```
  - (MKOverlayView *)mapView:(MKMapView *)sender
              viewForOverlay:(id <MKOverlay>)overlay;
  ```

MKOverlayView

- MKOverlayView subclasses must be able to draw the overlay

```
- (void)drawMapRect:(MKMapRect)mapRect
          zoomScale:(MKZoomScale)zoomScale
          inContext:(CGContextRef)context;
```

  This is not quite like drawRect: (because you'll notice that you are provided the context).
  But you will still use CoreGraphics to draw (this method must be thread-safe, by the way).
  Also notice that the rectangle to draw is in map coordinates, not view coordinates.

- Converting to/from map points/rects from/to view coordinates

```
- (MKMapPoint)mapPointForPoint:(CGPoint)point;
- (MKMapRect)mapRectForRect:(CGRect)rect;
- (CGPoint)pointForMapPoint:(MKMapPoint)mapPoint;
- (CGRect)rectForMapRect:(MKMapRect)mapRect;
```

Demo

- MapKit

  Displaying Flickr photos on a map

Coming Up

- Next Lecture
  - Final Project Guidelines
  - MapKit Demo (if time did not permit today)
  - Persistence
- Friday Section
  - Time Profiler
  - How to measure the performance of your application