Stanford CS193p

Developing Applications for iOS
Fall 2011

Today

- View Controller Lifecycle
  When your controller hears about things and what you should do about it.
- Image View
  Kind of like "UILabel" for images (as opposed to text).
- Web View
  Complete browser in a view.
- Scroll View
  Provides a moving "viewport" on a rectangular area that has views (the scroll view's subviews) in it.
- Demo
  (time permitting)
  Dr. Pill's Website View Controller
  Imaginarium

View Controller Lifecycle

- View Controllers have a "Lifecycle"
  A sequence of messages is sent to them as they progress through it
- Why does this matter?
  You very commonly override these methods to do certain work
- We've talked about the first part of the lifecycle
  Creation
  This is done mostly either via a segue or storyboard's `instantiateViewControllerWithIdentifer:`.
  Because of this, we rarely (never?) override `UIViewController`'s designated initializer in iOS 5.
  `awakeFromNib` is an option, but we rarely do that either.
  There are better methods to initialize in ...

View Controller

- After instantiation and outlet-setting, `viewDidLoad` is called

  `- (void)viewDidLoad;`

  This is an exceptionally good place to put a lot of setup code.

  But be careful because the geometry of your view (its `bounds`) is not set yet!

  If you need to initialize something based on the geometry of the view, use the next method ...

- Just before the view appears on screen, you get notified

  `- (void)viewWillAppear:(BOOL)animated;`

  When this is called, your `bounds` has been set (via your `frame` by your `superview` or some such).

  Your view will probably only get "loaded" once, but it might appear and disappear a lot.

  So don't put something in this method that really wants to be in `viewDidLoad`.

  Otherwise, you might be doing something over and over unnecessarily.

  Use this to optimize performance by waiting until this method (i.e. just before view appears)
  to kick off an expensive operation (might have to put up a spinning "loading" icon though).

  Summary: this method is for geometry-related initialization and lazy execution for performance.

View Controller

- And you get notified when you will disappear off screen too

  This is where you put "remember what's going on" and cleanup code.

```
- (void)viewWillDisappear:(BOOL)animated
{
    [super viewWillDisappear:animated];  // call super in all the viewWill/Did... methods
    // let's be nice to the user and remember the scroll position they were at ...
    [self rememberScrollPosition];  // we'll have to implement this, of course
    // do some other clean up now that we've been removed from the screen
    [self saveDataToPermanentStore]; // maybe do in did instead?
    // but be careful not to do anything time-consuming here, or app will be sluggish
    // maybe even kick off a thread to do what needs doing here
}
```

- There are "did" versions of both of these methods too

```
- (void)viewDidAppear:(BOOL)animated;
- (void)viewDidDisappear:(BOOL)animated;
```

View Controller

- Frame changed? Here's a good place to layout subviews manually
  (if struts and springs are not enough)
  `- (void)view{Will,Did}LayoutSubviews;`
  Called any time a view's `frame` changed and its `subviews` were thus re-layed out.
  For example, autorotation.
  You can reset the `frames` of your `subviews` here (e.g. re-layout your Calculator!)
- Specific notification that rotation will/did happen

```
- (void)willRotateToInterfaceOrientation:(UIInterfaceOrientation)anOrientation
                                duration:(NSTimeInterval)seconds;
- (void)willAnimateRotationToInterfaceOrientation:(UIInterfaceOriention)orient
                                duration:(NSTimeInterval)seconds;
- (void)didRotateFromInterfaceOrientation:(UIInterfaceOrientation)anOrientation;
@property UIInterfaceOrientation interfaceOrientation;
```

  The property will have the current orientation when each of the above is called.
  Example use: doing anything expensive (e.g. an animation maybe?) in `will` and resume it in `did`.

View Controller

- In low-memory situations, `viewDidUnload` gets called …

  … after your UIViewController's `self.view` is set to `nil` (hopefully freeing up its heap usage).

  This can only happen if your MVC is not on-screen, of course!

  This rarely happens, but well-designed code will anticipate it.

  Even though your outlet pointers are `weak` and will probably get set to `nil` automatically,
  it is "good practice" to set them to `nil` here so that they are predictably `nil` when unloaded.

  For example, HappinessViewController's viewDidUnload should probably look like this:

```
- (void)viewDidUnload
{
    self.faceView = nil;
}
```

  If the UIViewController goes back on-screen, `viewDidLoad` (etc.) will be called again.

View Controller Initialization

- Creating a UIViewController from a .xib file
  This is the old, iOS 4 way. Not covered in this class.
  You create a .xib file with the same name as your UIViewController subclass.
  Then use alloc/init to create one.
  Designated initializer (only if you need to override it, use init otherwise):
  `- (id)initWithNibName:(NSString *)nibName bundle:(NSBundle *)bundle;`
- Creating a UIViewController's UI in code (no .xib, no storyboard)
  Override the method - (void)loadView and set self.view to something.
  This is either/or with storyboards/.xibs.
  Do NOT implement loadView if you use a storyboard/.xib to create the UIViewController.
  Do NOT set self.view anywhere else besides in loadView.
  Do NOT implement loadView without setting self.view (i.e. you must set self.view in loadView).

View Controller Initialization

- Avoid `awakeFromNib` if possible

  It is an acceptable place to initialize stuff for a `UIViewController` from a storyboard/`.xib`.

  But try to put stuff in `viewDidLoad`, `viewWillAppear:` or the segue preparation code instead.

UIView's frame

- Who's responsible for setting a UIView's frame?
  Answer: The object that puts the UIView in a view hierarchy.
- In Interface Builder, you set all view's frames graphically
  You do this by dragging on the little handles.
- What about the frame passed to initWithFrame:?
  If you're putting it into a view hierarchy right away, pick the appropriate frame.
  If you are not, then it doesn't really matter what frame you choose (but avoid CGRectZero).
  The code that eventually DOES put you in a view hierarchy will have to set the frame.
- Setting frames in viewDidLoad
  Recall that your final bounds are not set in viewDidLoad.
  If you create views in code in viewDidLoad, pick sensible frames based on the view's bounds then.
  But be sure to set struts/springs (UIView's autoresizingMask property).
  Think of adding something in viewDidLoad as the same as laying it out in Xcode.
  In both cases, you have to anticipate that the top-level view's bounds will be changed.

UIImageView

- A UIView subclass which displays a UIImage
  We covered how to create a UIImage in the lecture on Views (lecture 4).
- How to set the UIImageView's UIImage
  `- (id)initWithImage:(UIImage *)image; // will set its frame's size to match image's size`
  `@property (nonatomic, strong) UIImage *image; // will not adjust frame size`
- Remember UIView's contentMode property?
  (e.g. UIViewContentModeRedraw/Top/Left, etc., ScaleToFit is the default)
  Determines where the image appears in the UIImageView's bounds and whether it is scaled.
- Other features
  Highlighted image
  Sequence of images forming an animation

UIWebView

- A full internet browser in a UIView
  Can use it not only to take your users to websites, but to display PDFs, for example.
- Built on WebKit
  An open source HTML rendering framework (started by Apple).
- Supports JavaScript
  But limited to 5 seconds or 10 megabytes of memory allocation.
- Property to control whether page is scaled to fit the UIView
  `@property (nonatomic) BOOL scalesPagesToFit;`
  If `YES`, then page will be squished or stretched to fit the width of the UIView.
  If `NO`, the page will be its natural size and the user cannot zoom inside it.
- Property to get the scroll view it uses
  `@property (nonatomic, readonly, strong) UIScrollView *scrollView;`
  Can now set properties in the scroll view to control behavior.

UIWebView

- Three ways to load up HTML …

```
- (void)loadRequest:(NSURLRequest *)request;
- (void)loadHTMLString:(NSString *)string baseURL:(NSURL *)baseURL;
- (void)loadData:(NSData *)data
        MIMEType:(NSString *)MIMEtype
textEncodingName:(NSString *)encodingName
         baseURL:(NSURL *)baseURL;
```

- We'll talk about NSURLRequest in a moment
- Base URL is the "environment" to load resources out of

  i.e., it's the base URL for relative URL's in the data or HTML string.
- MIME type says how to interpret the passed-in data

  Multimedia Internet Mail Extension

  Standard way to denote file types (like PDF).

  Think "e-mail attachments" (that's where the name MIME comes from).

UIWebView

- NSURLRequest

```
+ (NSURLRequest *)requestWithURL:(NSURL *)url;
+ (NSURLRequest *)requestWithURL:(NSURL *)url
                     cachePolicy:(NSURLRequestCachePolicy)policy
                 timeoutInterval:(NSTimeInterval)timeoutInterval;
```

- NSURL

  Basically like an `NSString`, but enforced to be "well-formed."

  Examples: `file://...` or `http://...`

  In fact, it is the recommended way to specify a file name in the iOS API.

```
+ (NSURL *)urlWithString:(NSString *)urlString;
+ (NSURL *)fileURLWithPath:(NSString *)path isDirectory:(BOOL)isDirectory;
```

- NSURLRequestCachePolicy

  Ignore local cache; ignore caches on the internet; use expired caches;
  use cache only (don't go out onto the internet); use cache only if validated

UIScrollView

Adding subviews to a normal UIView …

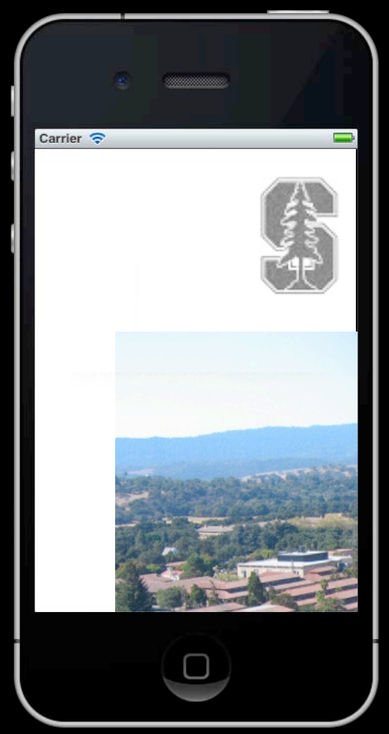
```
subview.frame = ...;
[view addSubview:subview];
```

Adding subviews to a normal UIView …

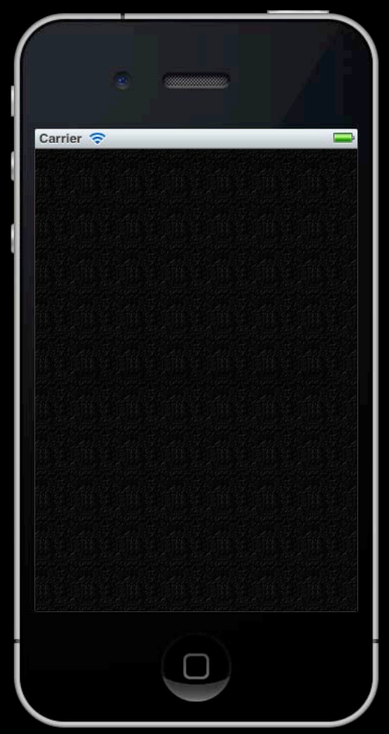
```
subview.frame = ...;
[view addSubview:subview];
```

Adding subviews to a normal UIView …

```
subview.frame = ...;
[view addSubview:subview];
```

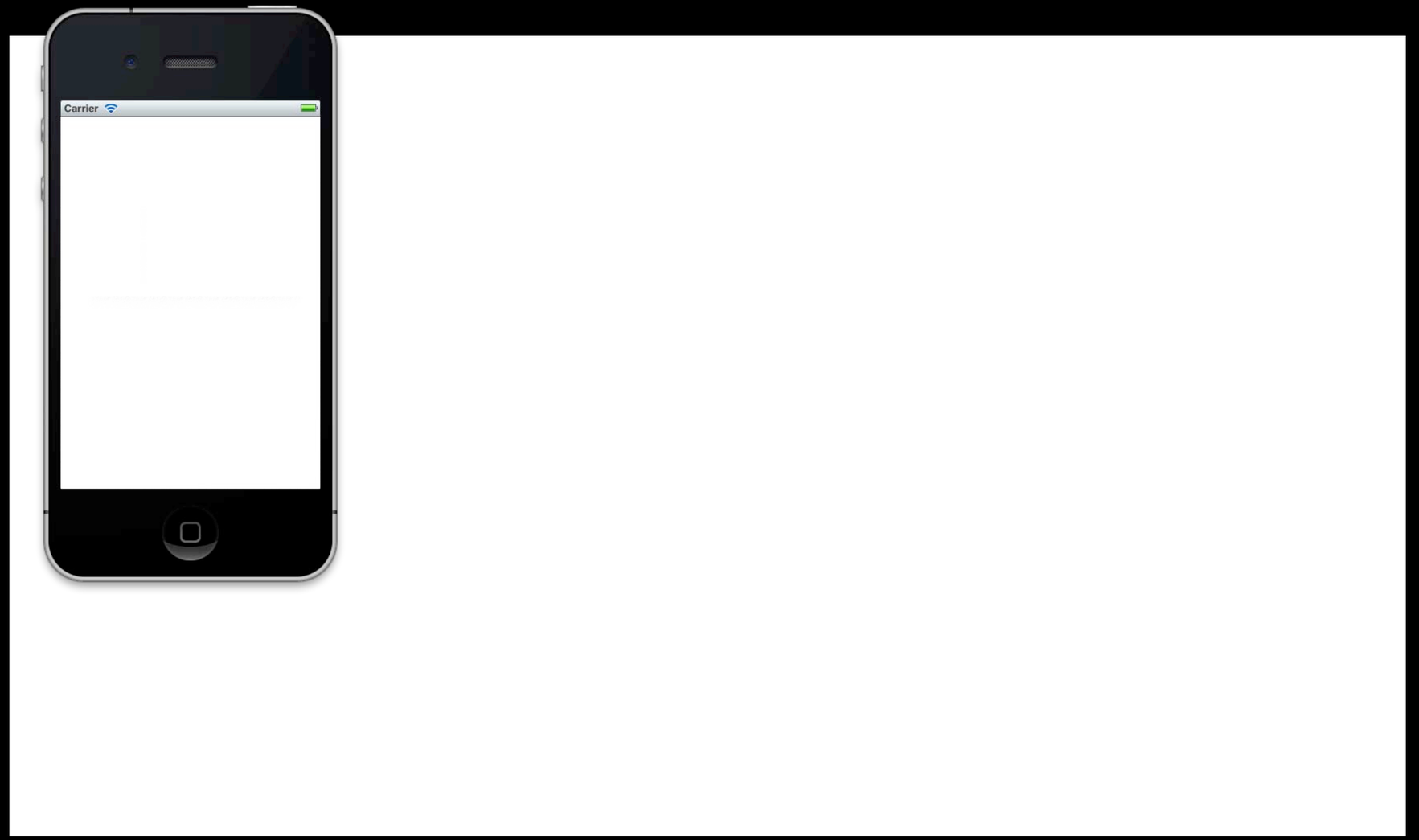


Adding subviews to a UIScrollView ...



Adding subviews to a UIScrollView …

```
scrollView.contentSize = CGSizeMake(3000, 2000);
```

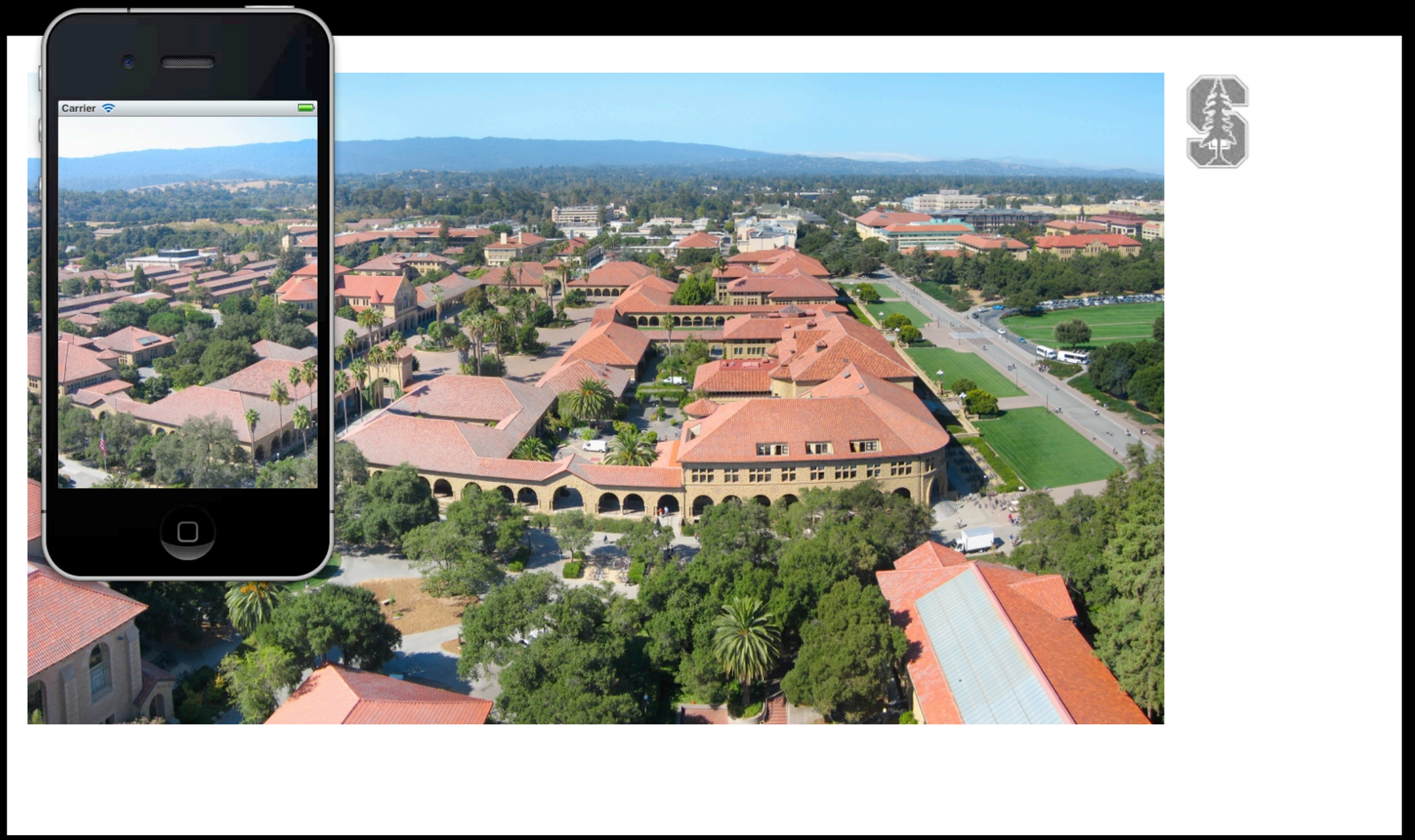


Adding subviews to a UIScrollView …

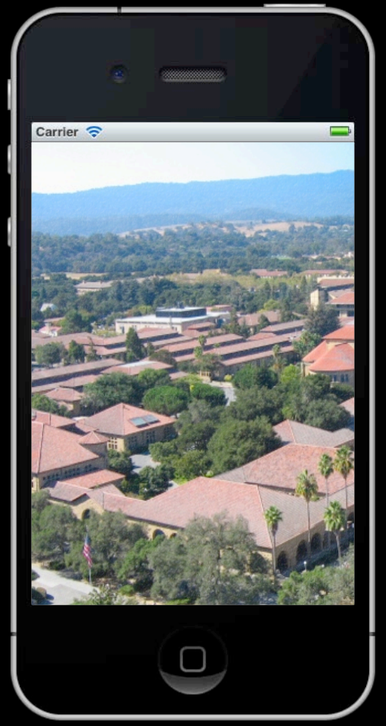
```
scrollView.contentSize = CGSizeMake(3000, 2000);
subview1.frame = CGRectMake(2700, 100, 120, 180);
[view addSubview:subview];
```

Adding subviews to a UIScrollView …

```
scrollView.contentSize = CGSizeMake(3000, 2000);
subview2.frame = CGRectMake(50, 100, 2500, 1600);
[view addSubview:subview];
```

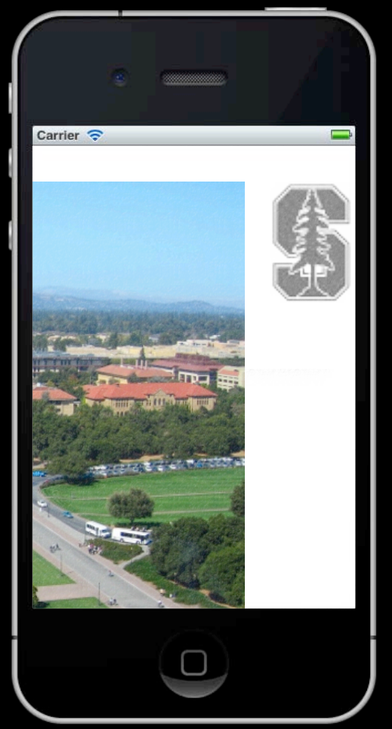


Adding subviews to a UIScrollView …

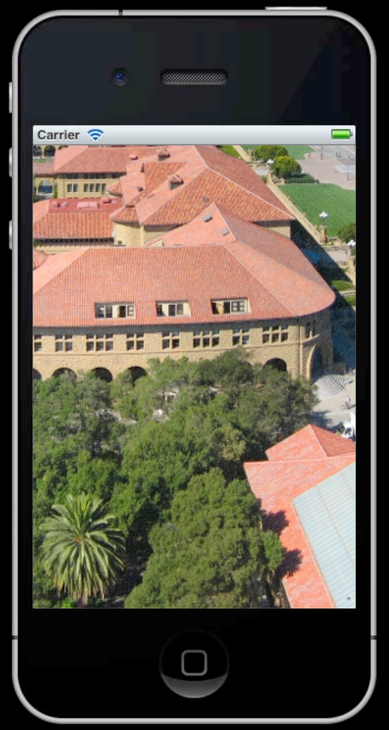


Adding subviews to a UIScrollView ...

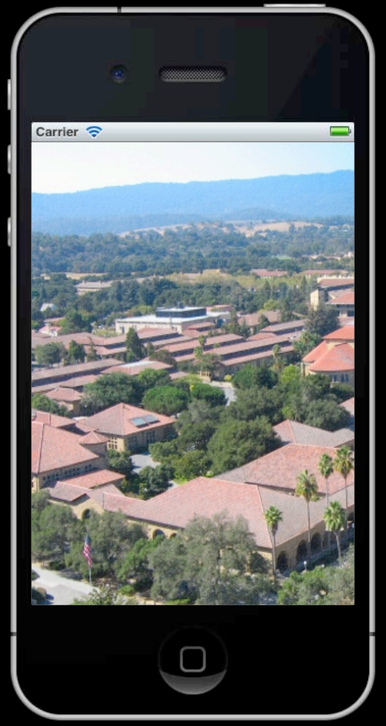
Adding subviews to a UIScrollView …



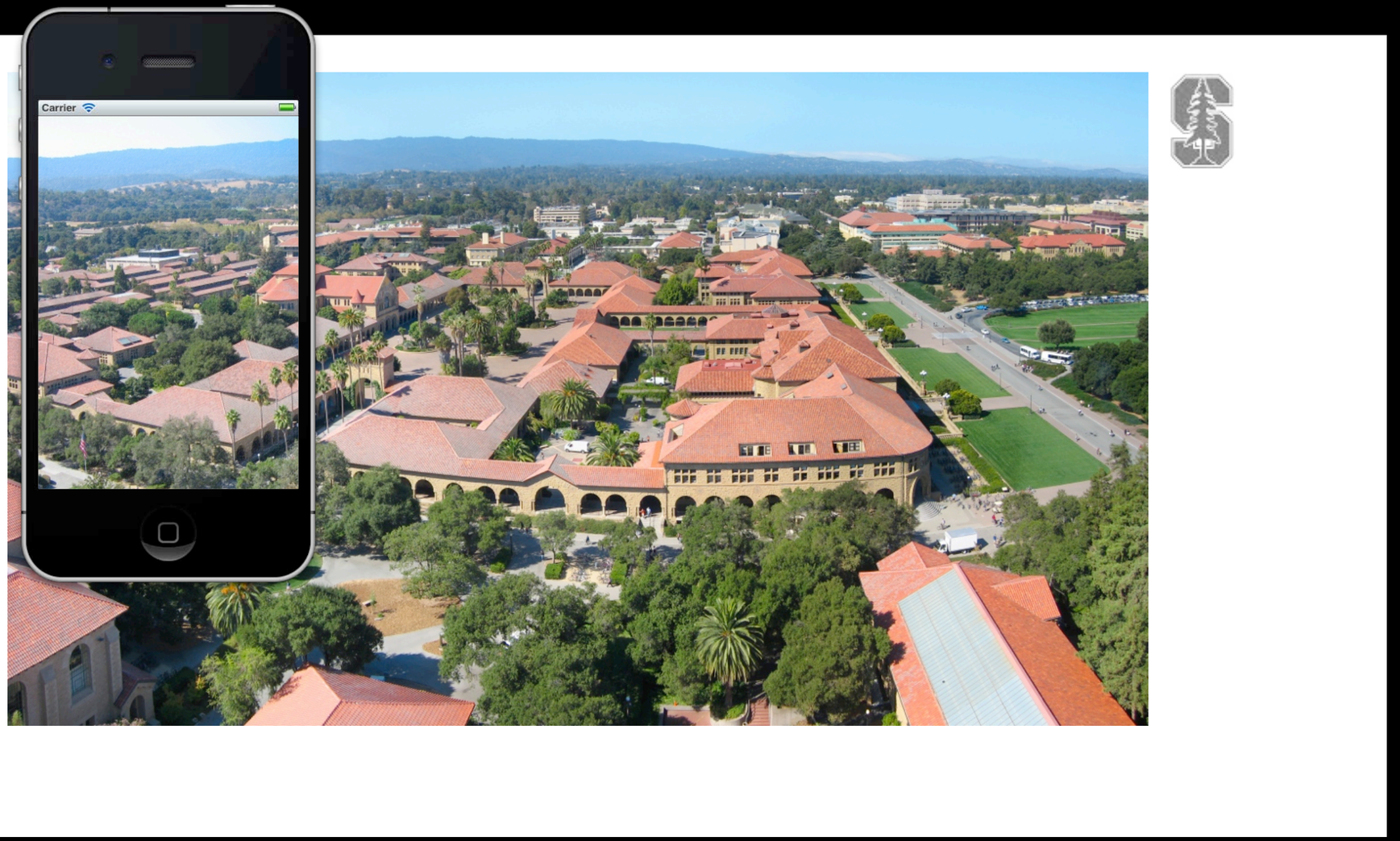
Adding subviews to a UIScrollView …



Adding subviews to a UIScrollView …



Positioning subviews in a UIScrollView ...



Positioning subviews in a UIScrollView …

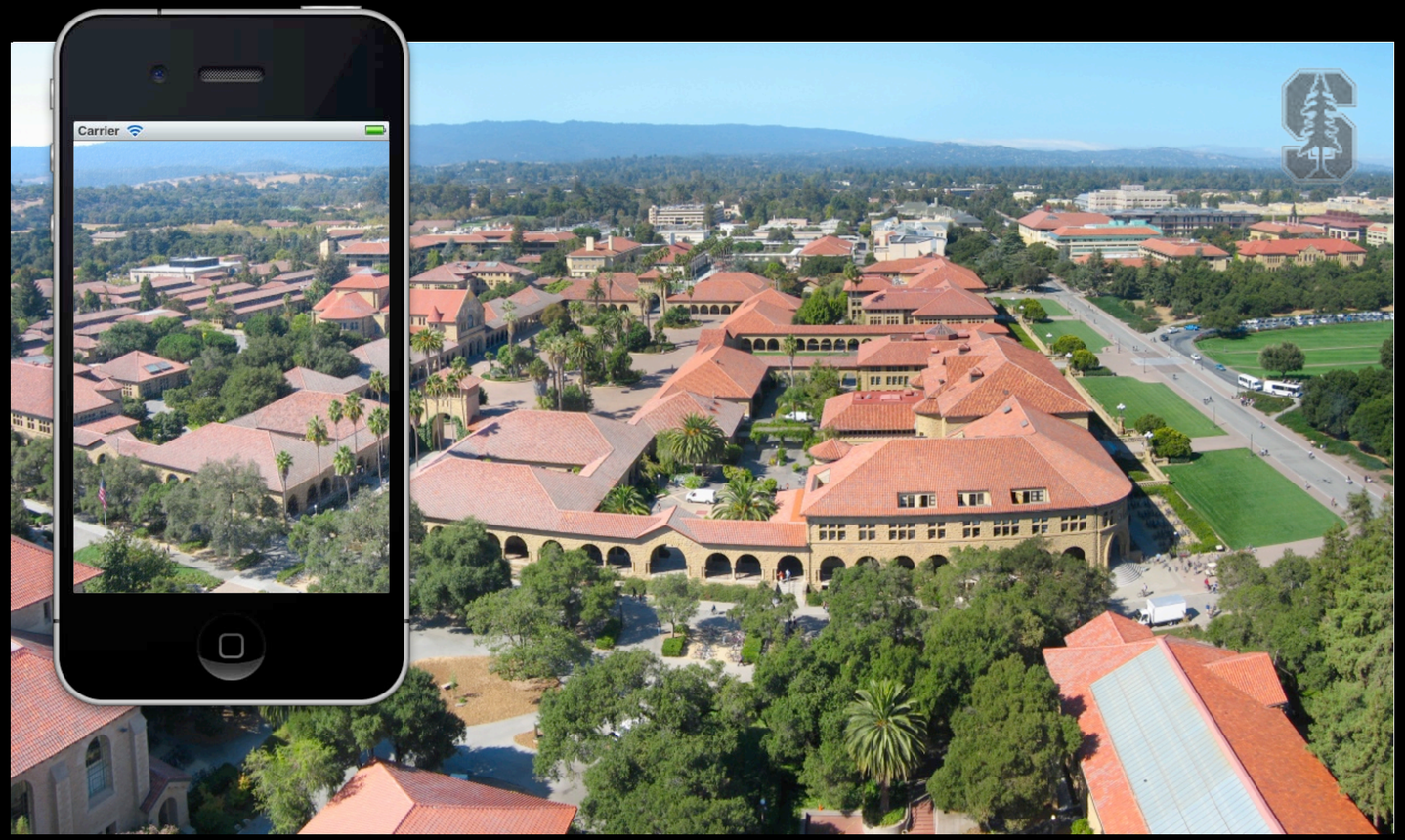
```
subview1.frame = CGRectMake(2250, 50, 120, 180);
```

Positioning subviews in a UIScrollView …

```
subview2.frame = CGRectMake(0, 0, 2500, 1600);
```

Positioning subviews in a UIScrollView …

```
subview2.frame = CGRectMake(0, 0, 2500, 1600);
scrollView.contentSize = CGSizeMake(2500, 1600);
```



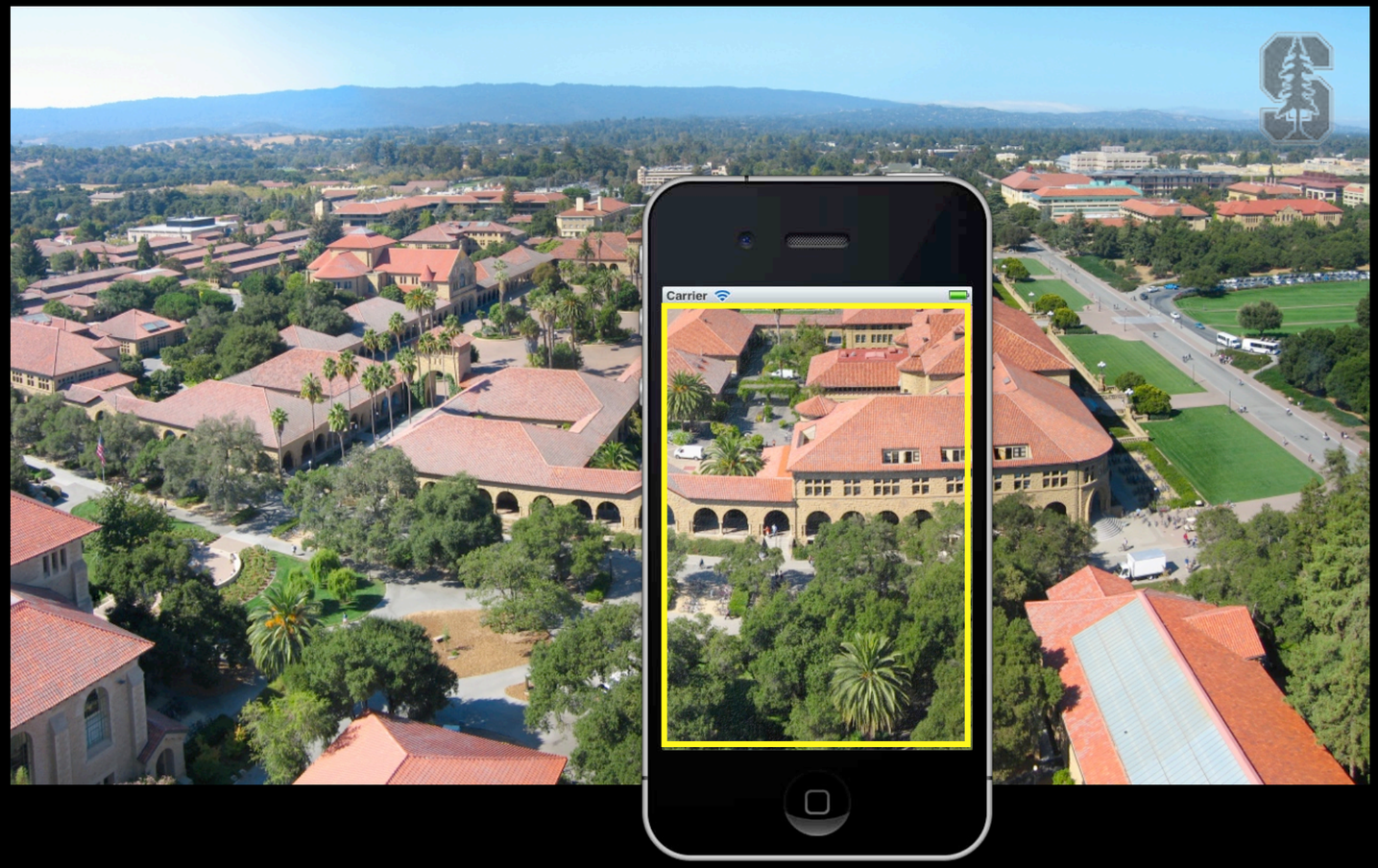
Upper left corner of currently-showing area

```
CGPoint upperLeftOfVisible = scrollView.contentOffset;
```

In content area's coordinates.

Visible area of a scroll view's subview in that view's coordinates

`scrollView.bounds`



Visible area of a scroll view's subview in that view's coordinates

```
CGRect visibleRect = [scrollView convertRect:scrollView.bounds toView:subview];
```

UIScrollView

- How do you create one?

  Just like any other UIView. Drag out in a storyboard or use alloc/initWithFrame:.

  Or select a UIView in your storyboard and choose "Embed In -> Scroll View" from Editor menu.

- Or add your "too big" UIView using addSubview:

```
UIImage *image = [UIImage imageNamed:@"bigimage.jpg"];
UIImageView *iv = [[UIImageView alloc] initWithImage:image]; // frame.size = image.size
[scrollView addSubview:iv];
```

  Add more subviews if you want.

  All of the subviews' frames will be in the UIScrollView's content area's coordinate system (that is, (0,0) in the upper left & width and height of contentSize.width & .height).

- Don't forget to set the contentSize!

  Common bug is to do the above 3 lines of code (or embed in Xcode) and forget to say:

```
scrollView.contentSize = imageView.bounds.size;
```

UIScrollView

- Scrolling programmatically

```
- (void)scrollRectToVisible:(CGRect)aRect animated:(BOOL)animated;
```

- Other things you can control in a scroll view

Whether scrolling is enabled.

Locking scroll direction to user's first "move".

The style of the scroll indicators (call `flashScrollIndicators` when your scroll view appears).

Whether the actual content is "inset" from the scroll view's content area (`contentInset` property).

UIScrollView

- Zooming

  All UIView's have a property (transform) which is an affine transform (translate, scale, rotate).
  Scroll view simply modifies this transform when you zoom.
  Zooming is also going to affect the scroll view's contentSize and contentOffset.

- Will not work without minimum/maximum zoom scale being set

```
scrollView.minimumZoomScale = 0.5;    // 0.5 means half its normal size
scrollView.maximumZoomScale = 2.0;    // 2.0 means twice its normal size
```

- Will not work without delegate method to specify view to zoom

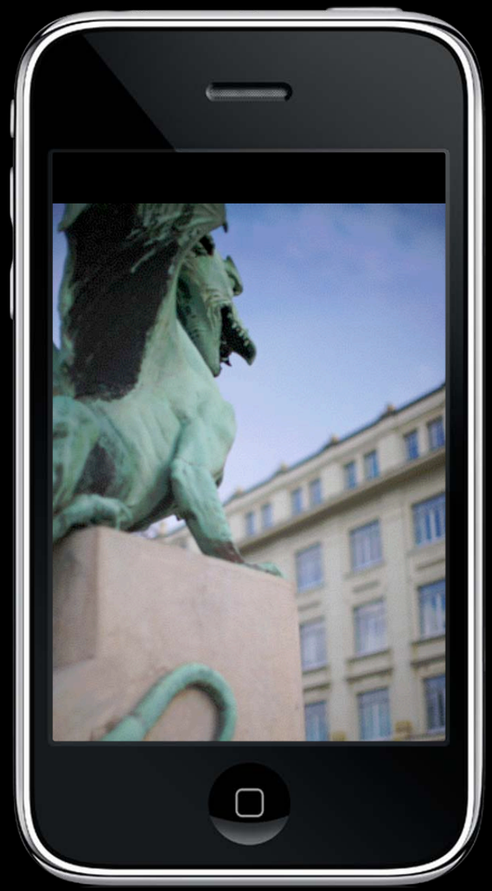
```
- (UIView *)viewForZoomingInScrollView:(UIScrollView *)sender;
```

  If your scroll view only has one subview, you return it here. More than one? Up to you.

- Zooming programatically

```
@property (nonatomic) float zoomScale;
- (void)setZoomScale:(float)scale animated:(BOOL)animated;
- (void)zoomToRect:(CGRect)zoomRect animated:(BOOL)animated;
```

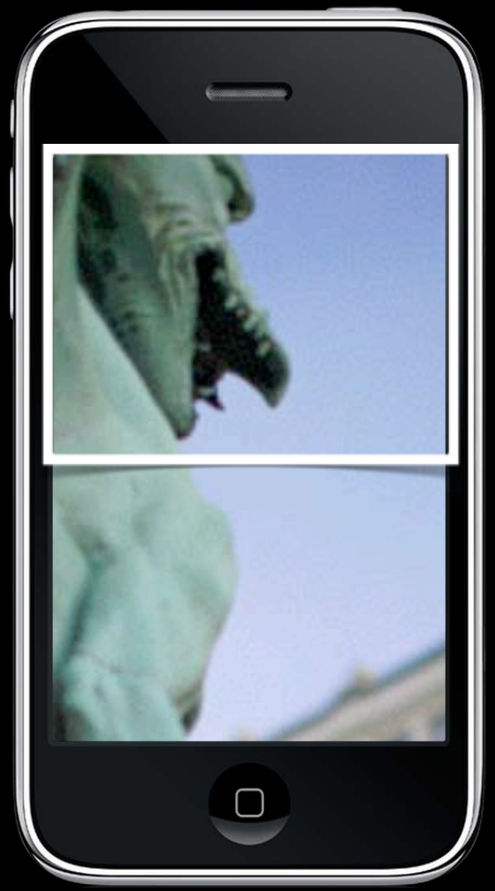
```
scrollView.zoomScale = 1.2;
```



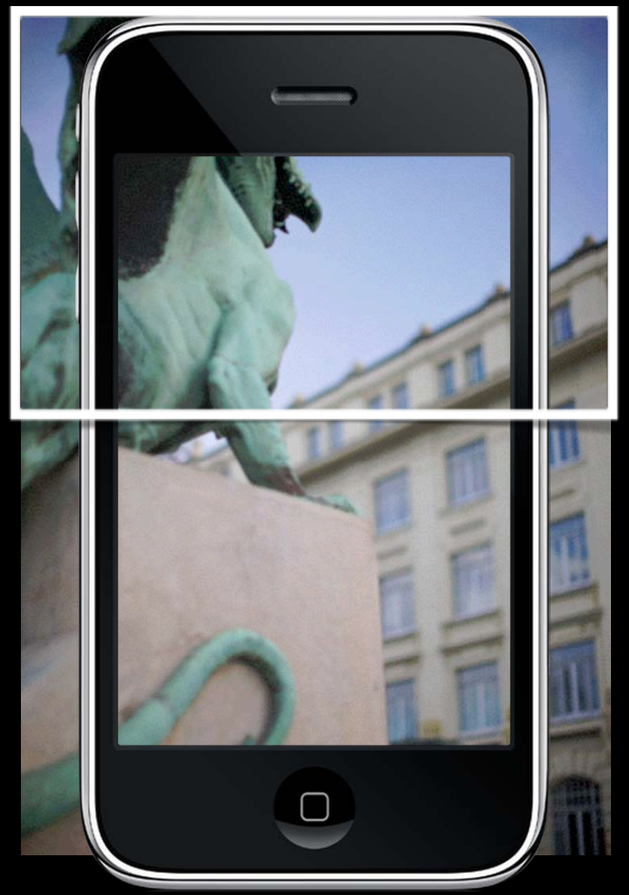
```
scrollView.zoomScale = 1.0;
```

```
scrollView.zoomScale = 1.2;
```

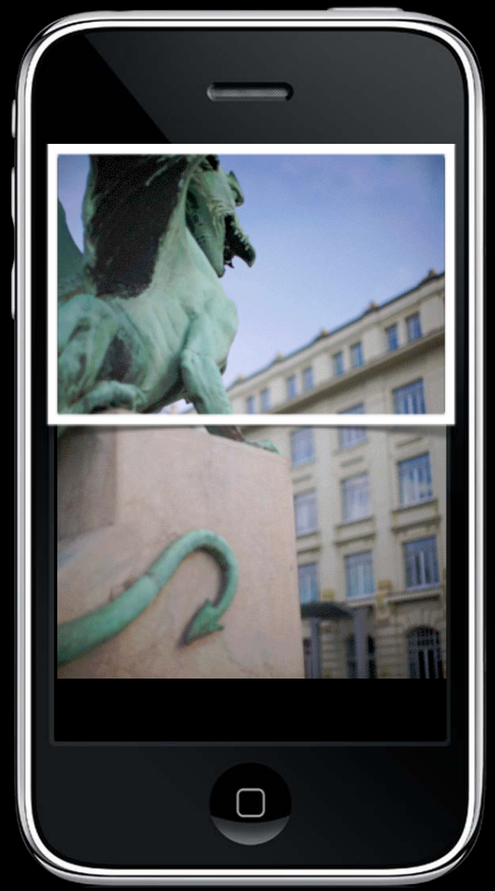
```
- (void)zoomToRect:(CGRect)rect animated:(BOOL)animated;
```



```
- (void)zoomToRect:(CGRect)rect animated:(BOOL)animated;
```



```
- (void)zoomToRect:(CGRect)rect animated:(BOOL)animated;
```



```
- (void)zoomToRect:(CGRect)rect animated:(BOOL)animated;
```

UIScrollView

- Lots and lots of `delegate` methods!

  The scroll view will keep you up to date with what's going on.

- Example: `delegate` method will notify you when zooming ends

```
- (void)scrollViewDidEndZooming:(UIScrollView *)sender
                       withView:(UIView *)zoomView  // from delegate method above
                        atScale:(CGFloat)scale;
```

  If you redraw your view at the new `scale`, be sure to reset the affine transform back to identity.

Demo

- Dr. Pill's Website

  Quick look at that UIWebView-using code.

- Imaginarium

  Simple UIImageView embedded inside a UIScrollView.

  Watch for ...

  View Controller Lifecycle method viewDidLoad.

  How we have to set the contentSize of the UIScrollView.

  How we have to set the frame of the UIImageView.

Coming Up

- Next Lecture
  Table View
- Section Tomorrow
  `AVFoundation` framework - Capturing and manipulating images.