Stanford CS193p

Developing Applications for iPhone 4, iPod Touch, & iPad

Fall 2010

Today

- Setting a `UIView`'s `frame`
  Who's responsible for this?
- `UIImageView`
- `UIWebView`
- `UIScrollView`
- Demo
  Imaginarium

UIView's frame

- Who's responsible for setting a UIView's frame?
  The object that puts the UIView in a view hierarchy.
- In Interface Builder, you set all view's frames
  You do this by dragging on the little handles.
- What about when you use alloc/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.
  This is because the code that eventually DOES put you in a view hierarchy will have to set it.
- When creating your view with alloc/initWithFrame: in loadView
  A good default to pick is [[UIScreen mainScreen] applicationFrame].
  This rectangle represents the part of the screen available to applications.
  Currently this is the entire screen minus the status bar at the top.
  This is what view controllers like UINavigation/TabBar/SplitViewController (likely) do.
  So you don't have to set their view's frame in application:didFinishLaunchingWithOptions:
  (even though you are adding their view to the view hierarchy with [window addSubview:...]).

UIImage

- Create by name

  Looks in your Resources folder from Xcode for a file with the given name …

  ```
  UIImage *image = [UIImage imageNamed:@“foo.jpg”];
  ```

- Create with data

  From the filesystem (we haven't talked about that yet!) …

  ```
  UIImage *image = [UIImage imageWithContentsOfFile:(NSString *)path];
  ```

  Or raw data (perhaps obtained from a network source like Flickr?) …

  ```
  UIImage *image = [[UIImage alloc] initWithData:(NSData *)data];
  ```

- How big is it?

  In points, not pixels …

  ```
  @property (readonly) CGSize size;
  ```

UIImageView

- What can you do with a UIImage?
  - Use it in your drawRect: (already saw this two weeks ago)
  - Ask a UIImageView to draw it
- UIImageView
  - Just a UIView, not unlike UILabel or even your custom graph view
  - Create it by dragging it out from Interface Builder
  - Or using code …

```
UIImageView *imageView = [[UIImageView alloc] initWithImage:(UIImage *)image];
```

- You can change the image in the UIImageView at any time

```
@property (retain) UIImage *image;
```

  - Note that setting the image after initialization does NOT modify the UIImageView's frame.

UIImageView

- UIImageView also has a highlighted option

```
@property BOOL highlighted;
@property (retain) UIImage *highlightedImage;
UIImageView *imageView = [[UIImageView alloc] initWithImage:(UIImage *)
                                                highlightedImage:(UIImage *)];
```

- And it can animate a sequence of images

```
@property (retain) NSArray *animationImages;  // of UIImage
@property (retain) NSArray *highlightedAnimationImages;
@property NSTimeInterval animationDuration;
@property NSInteger animationRepeatCount;
@property BOOL isAnimating;
- (void)startAnimating;
- (void)stopAnimating;
```

UIWebView

- An internet browser in a `UIView`
  Not just for displaying HTML (though it's good at that).
  Also can be used to display PDF's and other complex documents.
- Easy to connect back, reload, etc., buttons to it
- Based on WebKit
  An open source HTML rendering framework (started by Apple).
- Very easy to tell it to load up a page
- Can also use delegate to control/observe user's clicking
  Can prevent clicking through certain links or respond to user's navigation
- Supports JavaScript
  But limited to 5 seconds or 10 megabytes of memory allocation

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;
```

- 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://...`

  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

UIWebView

- Putting "browser" user-interface around a web view

```
@property (readonly) BOOL loading;
@property (readonly) BOOL canGoBack;
@property (readonly) BOOL canGoForward;
- (void)reload;
- (void)stopLoading;
- (void)goBack;
- (void)goForward;
```

- Controlling the display of the web view

```
@property BOOL scalesPageToFit;  // default is NO!

@property UIDataDetectorTypes dataDetectorTypes;
```

Can be `UIDataDetectorTypePhoneNumber/Link/Address/CalendarEvent`.
Will automatically open appropriate application if user clicks on these "detected" data elements.

UIWebView

- Delegate

```
- (BOOL)webView:(UIWebView *)sender
    shouldStartLoadWithRequest:(NSURLRequest *)request
    navigationType:(UIWebViewNavigationType)navigationType
```

- UIWebViewNavigationType

```
UIWebViewNavigationTypeLinkClicked
UIWebViewNavigationTypeFormSubmitted
```

- Other delegate notifications

```
- (void)webViewDidStartLoad:(UIWebView *)sender
- (void)webViewDidFinishLoad:(UIWebView *)sender
- (void)webView:(UIWebView *)sender didFailLoadWithError:(NSError *)error;
```

UIScrollView

- What to do when a view is just "too big" to fit on screen?
- Need to pan and zoom without having to implement gestures
- Enter `UIScrollView`
- A `UIView`
- Pans and zooms around a predefined `size` containing its `subviews`
- Two important subclasses of it: `UITextView`, `UITableView`

UIScrollView Examples

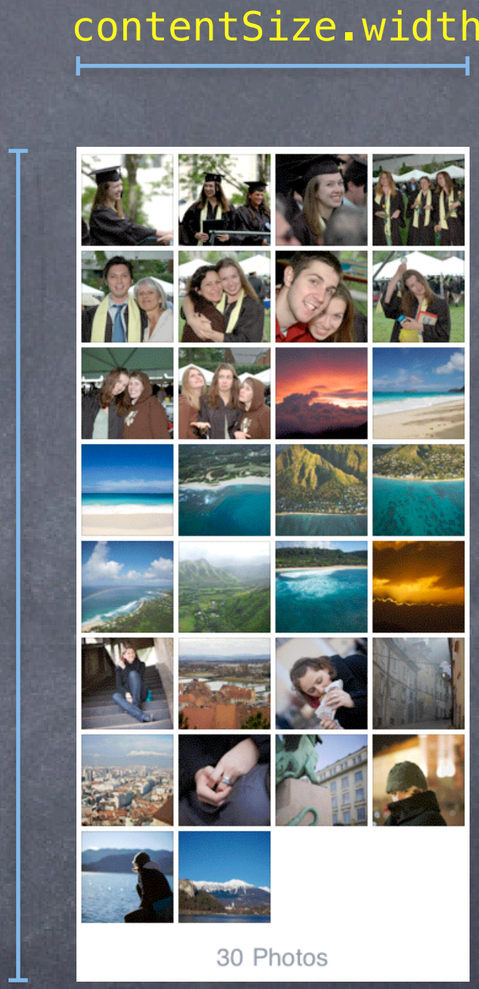
UIScrollView

```
@property CGSize contentSize;
```

The `frame` of each of the `UIScrollView`'s `subviews` is relative to this predefined space. A subview with a `frame` with an `origin` at `(0, 0)` would be in the upper left of this space.

Usually the `UIScrollView` only has one subview which fills the entire space, i.e. that subview's `frame` is usually:

```
origin = (0, 0)
size = scrollView.contentSize
```

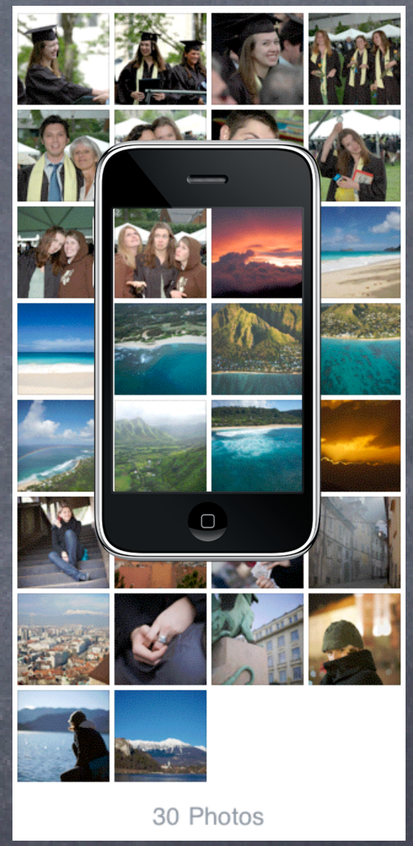


UIScrollView

```
@property CGSize contentSize;
```

contentSize.width

contentSize.height

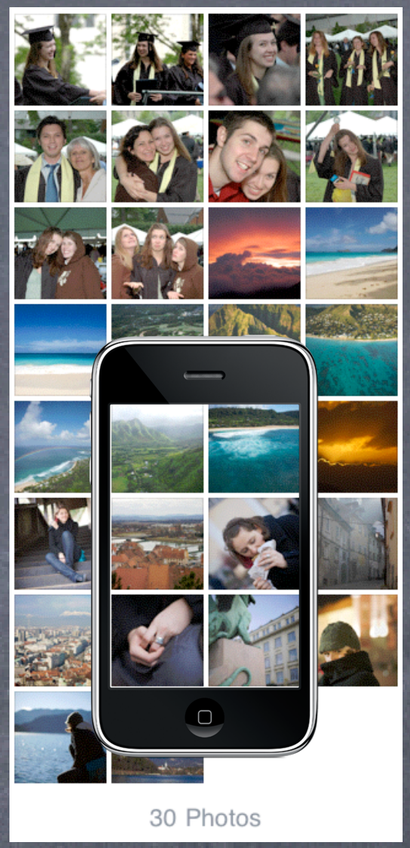


UIScrollView

```
@property CGSize contentSize;
```

contentSize.width

contentSize.height

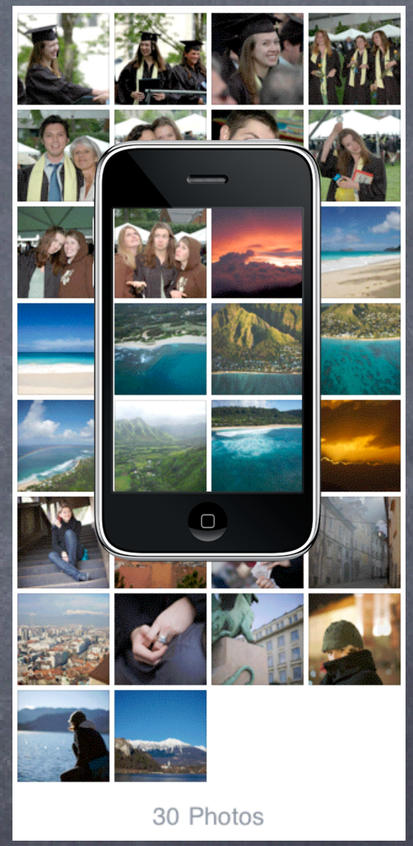


UIScrollView

```
@property CGSize contentSize;
```

contentSize.width

contentSize.height



UIScrollView

```
@property CGSize contentSize;
@property UIEdgeInsets contentInset;


struct {
   CGFloat top, bottom, left, right;
} UIEdgeInsets;
```

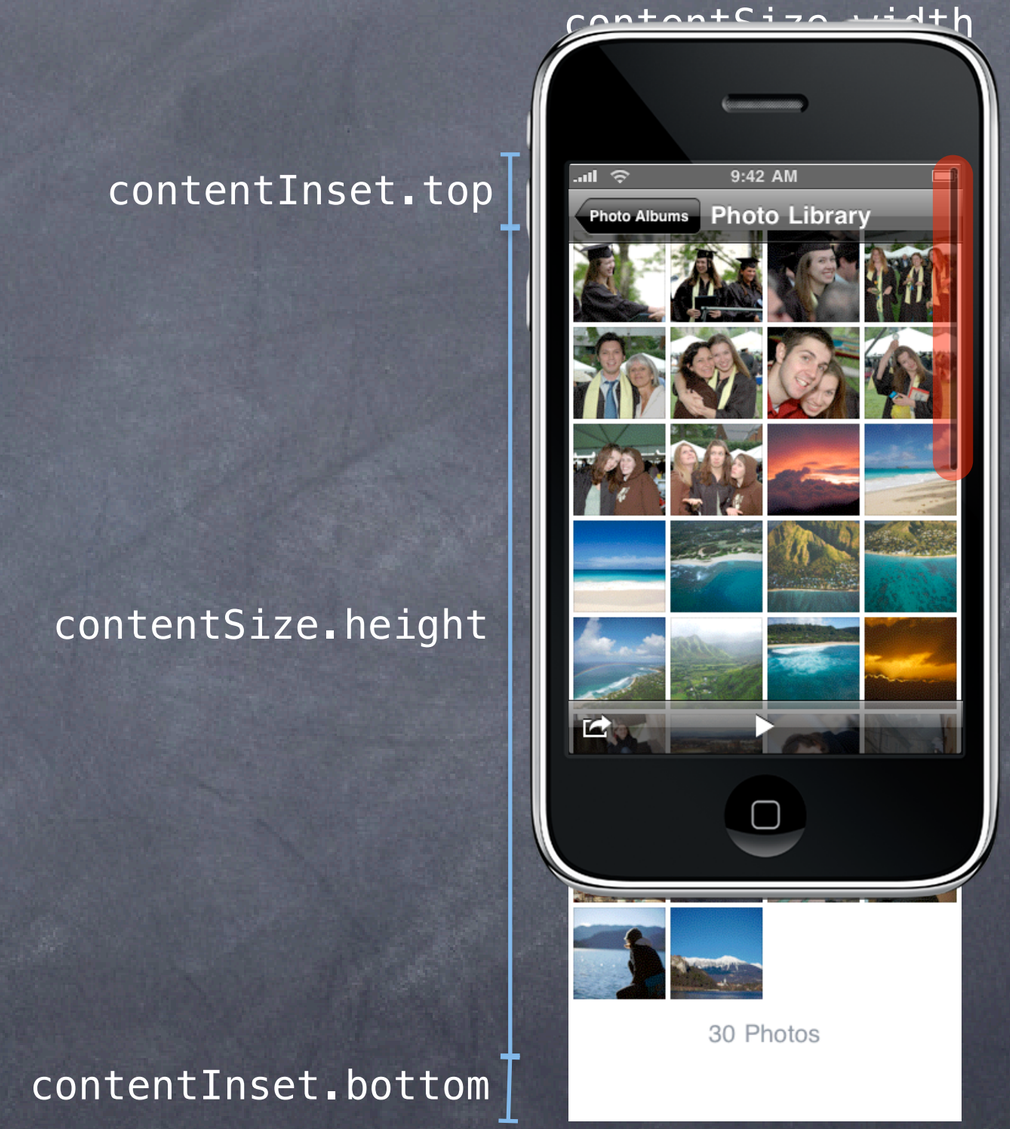
UIScrollView

```
@property CGSize contentSize;
@property UIEdgeInsets contentInset;


struct {
   CGFloat top, bottom, left, right;
} UIEdgeInsets;
```

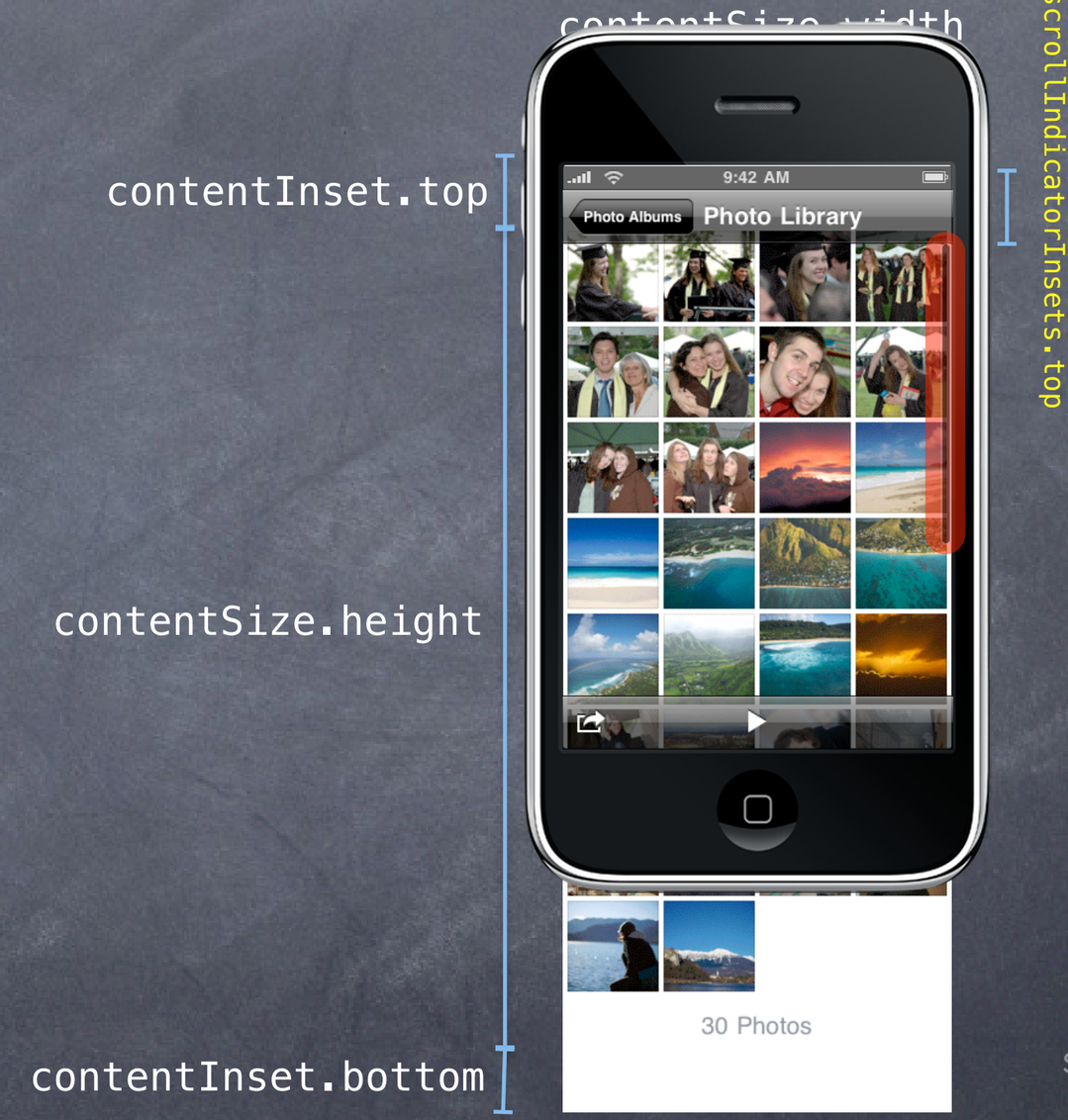
UIScrollView

```
@property CGSize contentSize;
@property UIEdgeInsets contentInset;
@property UIEdgeInsets scrollIndicatorInsets;
struct {
   CGFloat top, bottom, left, right;
} UIEdgeInsets;
```



UIScrollView

```
@property CGSize contentSize;
@property UIEdgeInsets contentInset;
@property UIEdgeInsets scrollIndicatorInsets;
struct {
   CGFloat top, bottom, left, right;
} UIEdgeInsets;
```

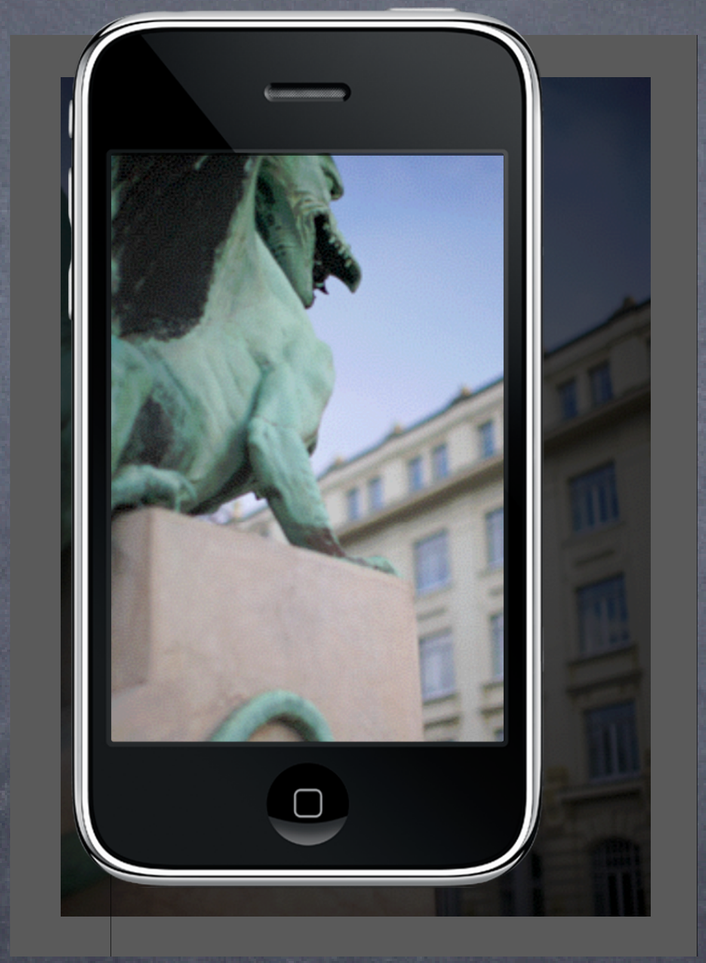


UIScrollView

```
@property CGSize contentSize;
@property UIEdgeInsets contentInset;
@property UIEdgeInsets scrollIndicatorInsets;
struct {
   CGFloat top, bottom, left, right;
} UIEdgeInsets;
```

UIScrollView

UIScrollView



```
@property CGSize contentOffset;
```

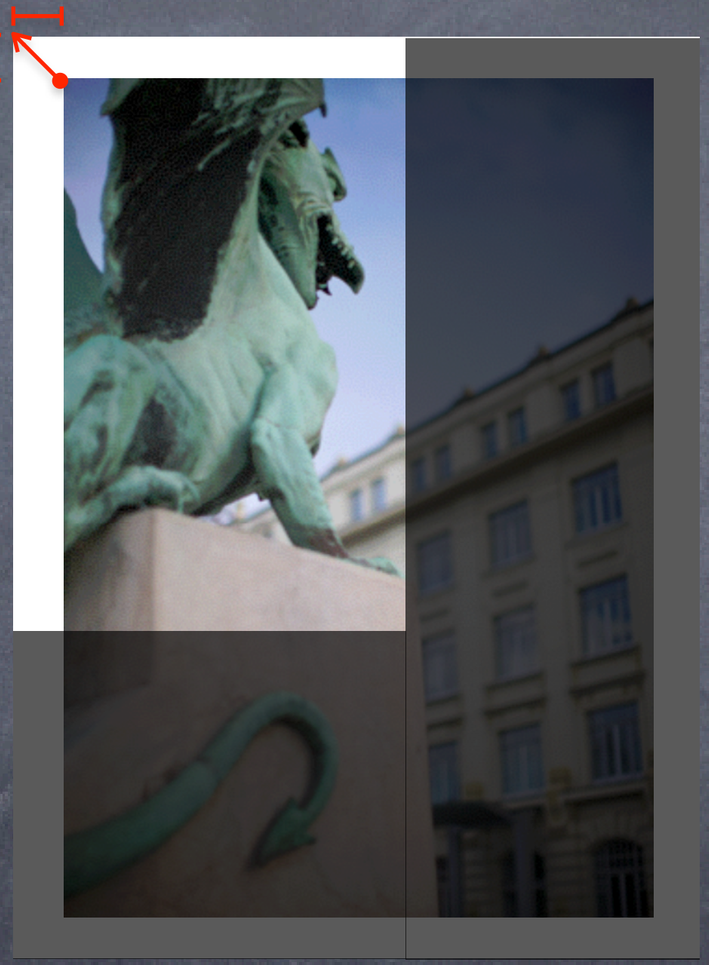
UIScrollView


```
@property CGSize contentOffset;
```

UIScrollView

contentOffset.x
(-contentInset.left)

contentOffset.y
(-contentInset.top)



```
@property CGSize contentOffset;
```

UIScrollView

- How do you create one?

  Just like any other `UIView` … in a `.xib` or with `alloc/initWithFrame:`

  Code example: Filling the entire screen with one (`application:didFinishLaunchingWithOptions:`)

```
CGRect frame = [[UIScreen mainScreen] applicationFrame];
UIScrollView *scrollView = [[UIScrollView alloc] initWithFrame:frame];
[window addSubview:scrollView];
```

- Just add your "too big" `UIView` as a subview (or multiple thereof)

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

- Setting the content size (the scroll view's scrollable area's size)

```
scrollView.contentSize = imageView.bounds.size;   // for example
```

UIScrollView

- What else is there to set in a scroll view besides positioning?

  In other words, what else besides `contentOffset`, `contentSize`, `contentInset`, etc.?

- Bouncing (when user scrolls/zooms "too far" it bounces back)

- Constraining scrolling

```
@property BOOL scrollEnabled;           // whether scrolling is currently allowed
@property BOOL directionalLockEnabled;  // whether user's scrolling start locks allowed scroll
```

- Display of scroll indicators

```
@property UIScrollViewIndicatorStyle indicatorStyle;
@property BOOL showsHorizontalScrollIndicator;    // vertical too, of course
- (void)flashScrollIndicators;  // you should call this when scroll view is revealed on screen
```

- Programmatic scrolling

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

UIScrollView

- Zooming

  All `UIView`'s have a property (`transform`) which is an affine transform (translate, scale, rotate).
  Scroll view just modifies this `transform` when you zoom.

- 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.

- Another delegate method will notify you when zooming ends

```
- (void)scrollViewDidEndZooming:(UIScrollView *)sender
                       withView:(UIView *)zoomView    // returned 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.

UIScrollView

- Programmatic zooming

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

Best explained graphically …

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

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

```
- (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

- Other delegate methods

```
- (void)scrollViewDidScroll:(UIScrollView *)sender;
- (void)scrollViewDidScrollToTop:(UIScrollView *)sender;
- (void)scrollViewWillBeginZoom:(UIScrollView *)sender withView:(UIView *)zoomView;
- (void)scrollViewDidEndScrollingAnimation:(UIScrollView *)sender;
- (void)scrollViewWillBeginDecelerating:(UIScrollView *)sender;
```

- Methods in `UIScrollView` to find out scrolling state

```
@property (readonly) BOOL zooming;
@property (readonly) BOOL dragging;
@property (readonly) BOOL tracking;
@property (readonly) BOOL decelerating;
```

Coming Up

- Demo
  - Imaginarium
  - `UIImageView`
  - `UIScrollView`
- Homework
  - Due tomorrow!
- Next Lecture
  - `UITableView`
  - `UITableViewController`