

iOS Mobile Development



Today

- 👁 Multithreading

Posting blocks on queues (which are then executed on other threads).

- 👁 UIScrollView

A “window” on an arbitrarily large content area that can be moved and zoomed.

- 👁 Demo

Imaginarium

- 👁 UITableView

(Time Permitting)

Data source-driven vertical list of views.

Multithreading

👁 Queues

Multithreading is mostly about “queues” in iOS.

Blocks are lined up in a queue (method calls can also be enqueued).

Then those blocks are pulled off the queue and executed on an associated thread.

👁 Main Queue

There is a very special queue called the “main queue.”

All UI activity **MUST** occur on this queue and this queue only.

And, conversely, non-UI activity that is at all time consuming must **NOT** occur on that queue.

We want our UI to be responsive!

Blocks are pulled off and worked on in the main queue only when it is “quiet”.

👁 Other Queues

Mostly iOS will create these for us as needed.

We’ll give a quick overview of how to create your own (but usually not necessary).

Multithreading

• Executing a block on another queue

```
dispatch_queue_t queue = ...;  
dispatch_async(queue, ^{ });
```

• Getting the main queue

```
dispatch_queue_t mainQ = dispatch_get_main_queue();  
NSOperationQueue *mainQ = [NSOperationQueue mainQueue]; // for object-oriented APIs
```

• Creating a queue (not the main queue)

```
dispatch_queue_t otherQ = dispatch_queue_create("name", NULL); // name a const char *!
```

• Easy mode ... invoking a method on the main queue

NSObject method ...

```
- (void)performSelectorOnMainThread:(SEL)aMethod  
    withObject:(id)obj  
    waitUntilDone:(BOOL)waitUntilDone;  
dispatch_async(dispatch_get_main_queue(), ^{ /* call aMethod */ });
```

Multithreading

👁 Example of an iOS API that uses multithreading

```
NSURLRequest *request = [NSURLRequest requestWithURL:[NSURL URLWithString:@"http://..."]];
NSURLConfiguration *configuration = ...;
NSURLSession *session = ...;
NSURLSessionDownloadTask *task;
task = [session downloadTaskWithRequest:request
                        completionHandler:^(NSURL *localfile, NSURLResponse *response, NSError *error) {
    /* want to do UI things here, can I? */
}];
[task resume];
```

`downloadTaskWithRequest:completionHandler:` will do its work (downloading that URL)
NOT in the main thread (i.e. it will not block the UI while it waits on the network).

The `completionHandler` block above might execute on the main thread (or not)
depending on how you create the `NSURLSession`.
Let's look at the two options (on or off the main queue) ...

Multithreading

👁 On the main queue ...

```
NSURLSession *session = [NSURLSession sessionWithConfiguration:configuration
                                                                delegate:nil
                                                                delegateQueue:[NSOperationQueue mainQueue]];

NSURLSessionDownloadTask *task;
task = [session downloadTaskWithRequest:request
                                   completionHandler:^(NSURL *localfile, NSURLResponse *response, NSError or *error) {
    /* yes, can do UI things directly because this is called on the main queue */
}];
[task resume];
```

Since the `delegateQueue` is the main queue, our `completionHandler` will be on the main queue. When the URL is done downloading, the block above will execute on the main queue. Thus we can do any UI code we want there. Of course, if you are doing non-UI things here, they'd best be quick (don't block main queue!).

Multithreading

👁 Off the main queue ...

```
NSURLSession *session = [NSURLSession sessionWithConfiguration:configuration]; // no delegateQueue
NSURLSessionDownloadTask *task;
task = [session downloadTaskWithRequest:request
                                completionHandler:^(NSURL *localfile, NSURLResponse *response, NSError *error) {
    dispatch_async(dispatch_get_main_queue(), ^{ /* do UI things */ });
    or [self performSelectorOnMainThread:@selector(doUIthings) withObject:nil waitUntilDone:NO];
}];
[task resume];
```

In this case, you can't do any UI stuff because the completionHandler is not on the main queue. To do UI stuff, you have to post a block (or call a method) back on the main queue (as shown).

UIScrollView



Adding subviews to a normal UIView ...

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



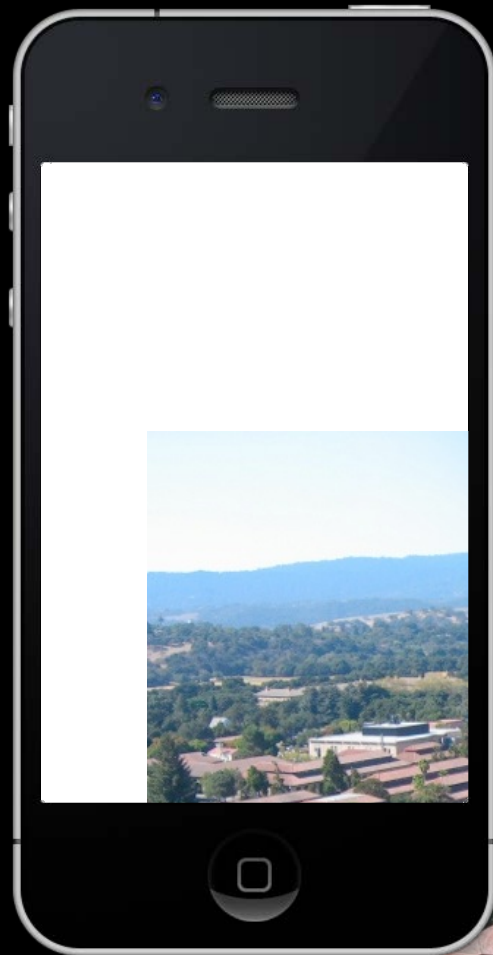
Adding subviews to a normal UIView ...

```
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:subview1];
```



Adding subviews to a UIScrollView ...

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



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 ...

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

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



Adding subviews to a UIScrollView ...

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



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);
```



Voilà!

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



Voilà!

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



Voilà!

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



Voilà!

```
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

`scrollView.bounds`



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

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

What's the difference? Might be scaled (due to zooming), for example.



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 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
```

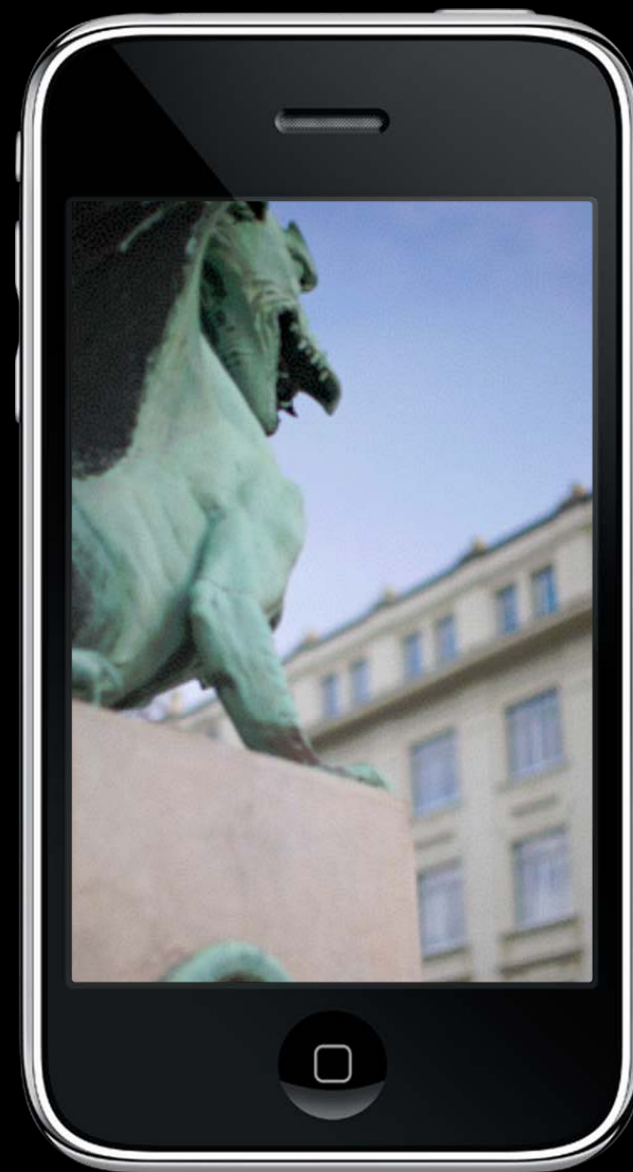
👁 Will not work without delegate method to specify view to zoom

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

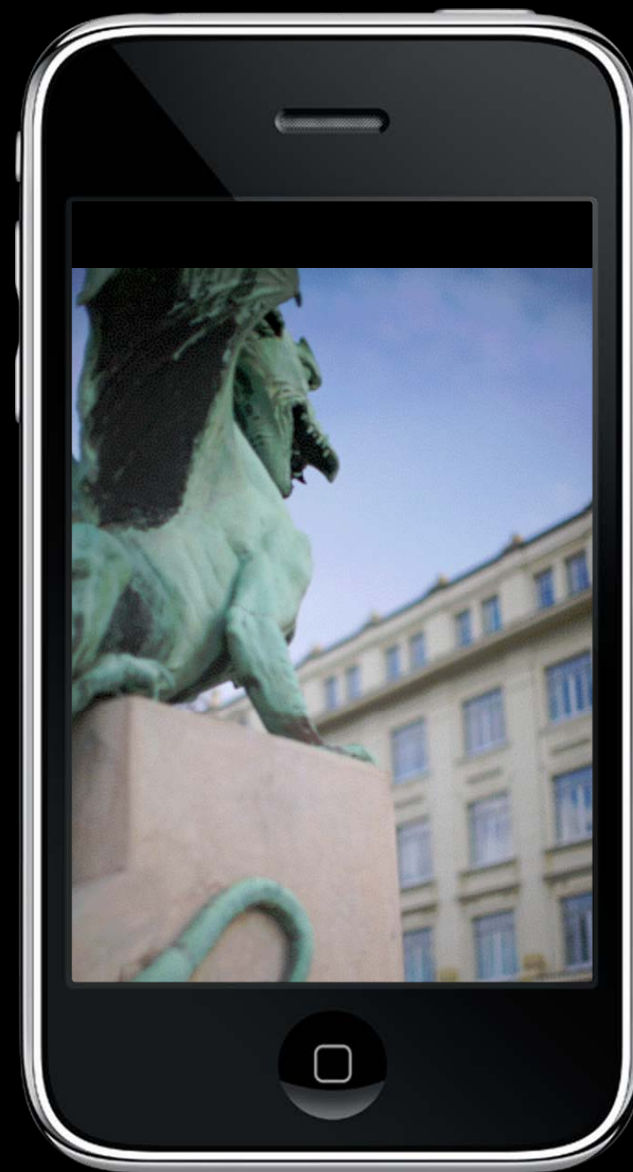
If your scroll view only has one subview, you return it here. More than one? Up to you.

👁 Zooming programmatically

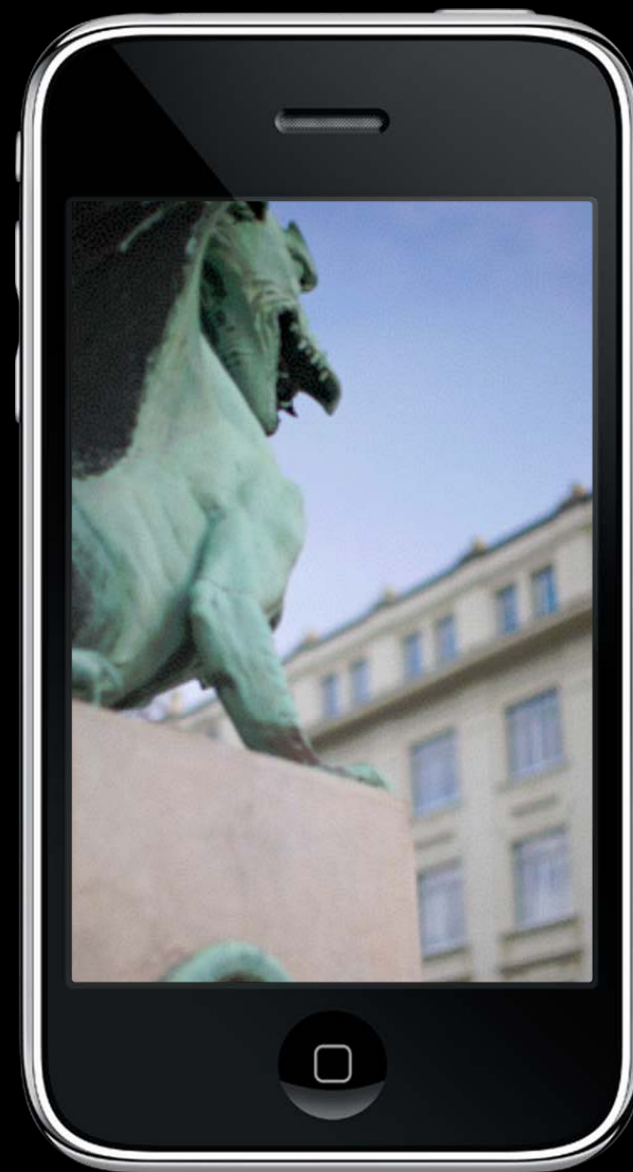
```
@property (nonatomic) float zoomScale;  
– (void)setZoomScale:(float)scale animated:(BOOL)animated;  
– (void)zoomToRect:(CGRect)zoomRect animated:(BOOL)animated;
```

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



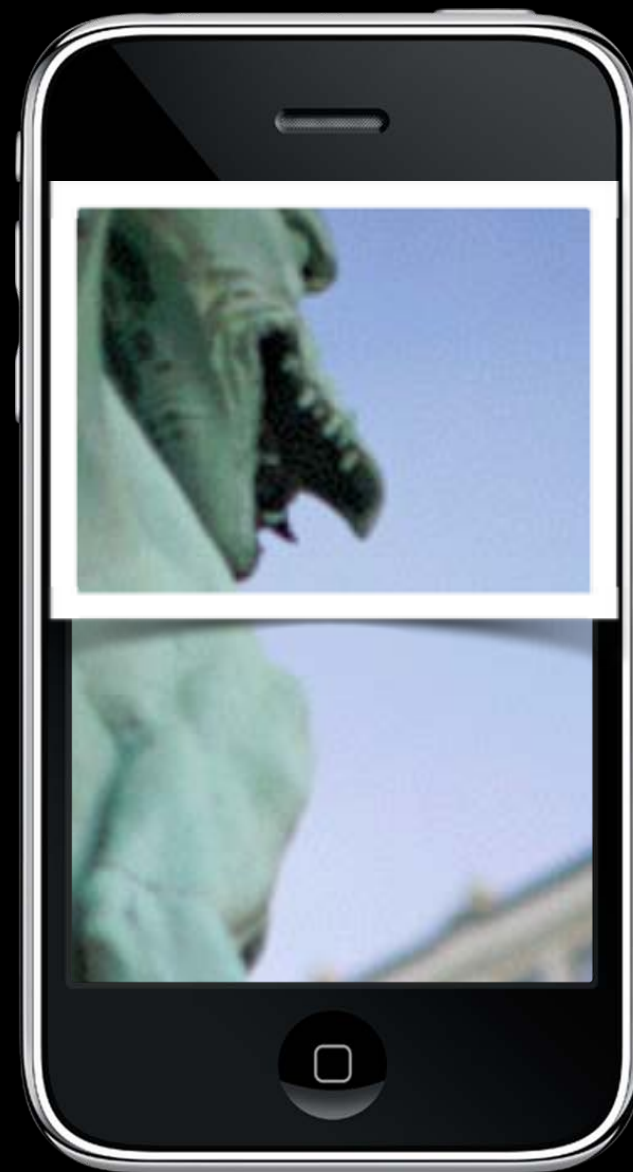
```
scrollView.zoomScale = 1.0;
```

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



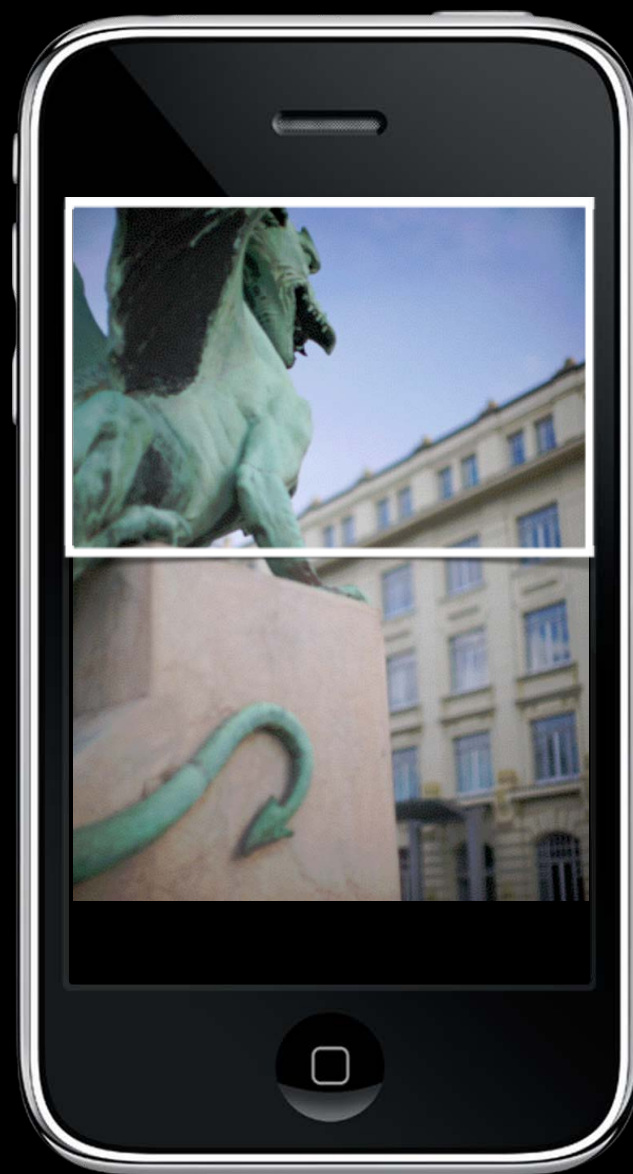
- (void)zoomToRect:(CGRect)rect animated:(BOOL)animated;



- (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 transform back to identity.

Demo

👁 Imaginarium

UIImageView inside a UIScrollView

Multithreaded download from a URL

UIActivityIndicatorView to show user that a download is in progress

Coming Up

More UITableView (with demo)
iPad

Core Data (Object-Oriented Database)