

HIT3329 / HIT8329
Creating Data Driven Mobile Applications

Lecture 7

Concurrent Operations, Events, and User Experience

Presented by Paul Chapman

Adjunct Lecturer F-ICT
Director, Long Weekend LLC
Swinburne University of Technology

Last Lecture Reviewed

1. Simple Networking
2. UI Concerns & Limiting Conditions
3. iOS App Lifecycle
4. Model-View-Controller (MVC)

Questions?

What's On For Today?

1. Libraries
2. Concurrent Operations
3. Run Loops & Timers
4. UI Feedback

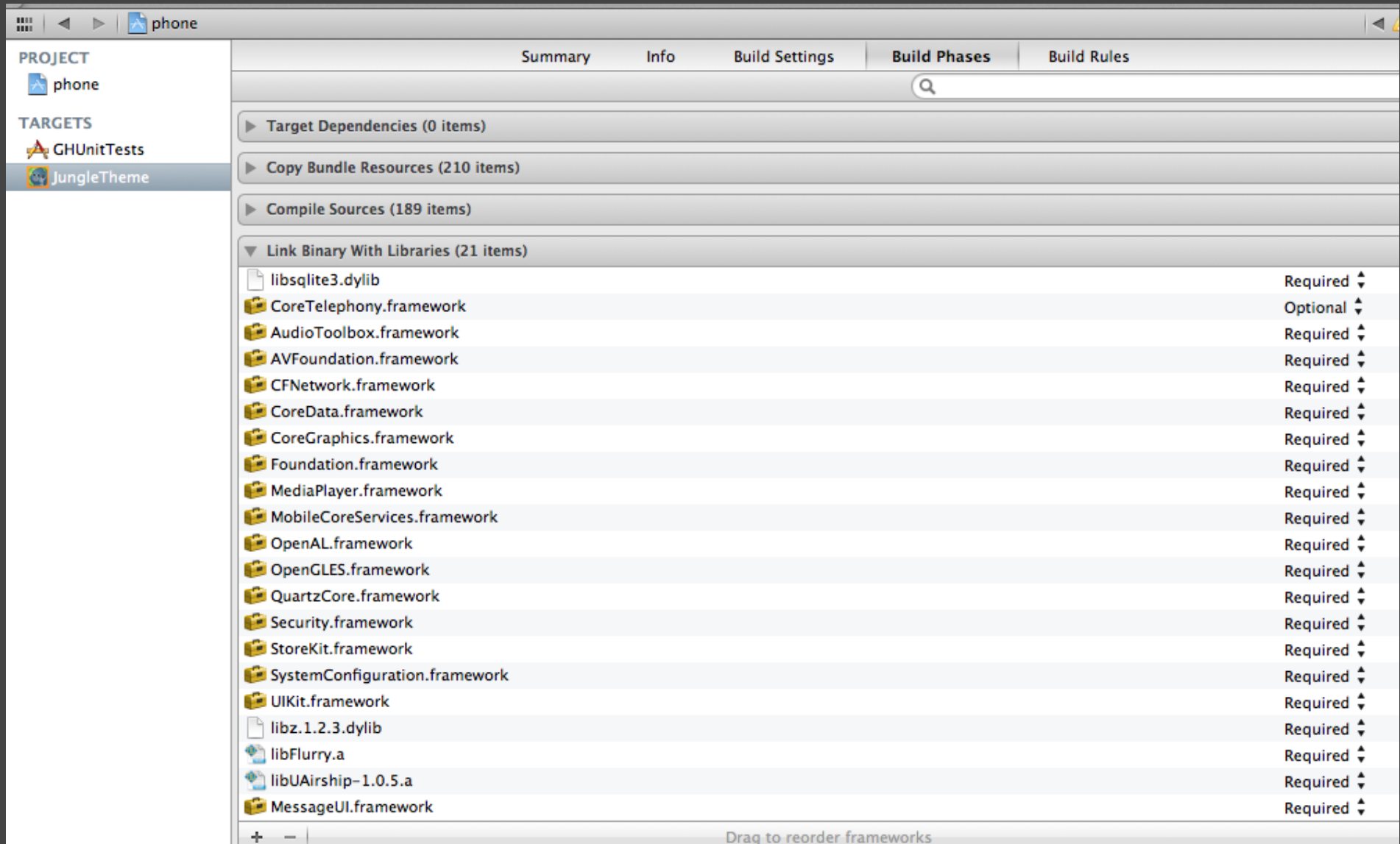
1.0 External Libraries

- Code written by others is often bundled into *libraries*
- Libraries help you with common tasks
> don't re-invent the wheel
- Usually free & public on the Internet, but not always

1.1 Apple Libraries

- Apple provides *Frameworks*, but they are usually quite low-level
- Some Cocoa Touch APIs require adding frameworks to Xcode project "Link Binary With Libraries" step (ex. `MediaPlayer.framework`)

1.2 Adding a Library in Xcode



1.3 Don't Do It Yourself, Use a Library

Angry Birds used:

- Cocos2D - graphics engine library
- Chipmunk - physics engine library

Rovio added a good game concept, original graphics and addictive game play. Yet it's highly likely there are more lines of code in these libraries than in their source code.

1.4 Where To Find Them

Many of these libraries' source code is hosted publicly:

- GitHub (github.com)
- Google Code (code.google.com)
- ... and more

1.5 How To Use External Libraries

- "Checkout" the repository if you're comfortable with source code management tools like git or svn.
- Or, just download the source code and add it to your Xcode project
- Can also be a binary file (.a)

1.5 Why This, Now?

- Last lecture we downloaded data from the Net in XML or JSON format
- To use XML or JSON in your app, you need to process it yourself
- Thankfully someone else has already done the hard work!

1.6 Data Formats I

Never heard of JSON or XML?



1.7 Data Formats II

- JSON and XML are the basic data interchange formats
- XML first codified in the late 1990s
- JSON is part of the JavaScript specification but only came into use in mid-2000s

1.8 Data Formats III

XML is somewhat human-readable:

```
<dinner type="classy">  
<entree>  
<salad style="chef" /> </entree>  
<mains>  
<steak cooktime="medium-rare"/>  
<bread />  
</mains>  
<dessert />  
</dinner>
```

1.9 Data Formats IV

JSON is rather more human-readable:

```
{  
  "firstName": "John",  
  "lastName": "Smith",  
  "age": 25,  
  "address": {  
    "streetAddress": "21 2nd Street",  
    "city": "New York",  
    "state": "NY",  
    "postalCode": "10021"  
  },  
}
```

1.10 json-framework for Objective-C

The json-framework library converts JSON data structures like this:

```
{ "firstName": "John", "lastName": "Smith" }
```

Into Objective-C collection objects like this:

```
[NSDictionary dictionaryWithObjectsAndKeys:  
    @"John",@"firstName",  
    @"Smith",@"lastName",nil];
```

See: <https://github.com/stig/json-framework/>

1.11 Problems with Data Processing

- **Problem:** Sometimes converting lots of JSON to NSObjects takes a long time to finish (>500 msec)
- **Solution:** ??
- **Hint:** Think back to last lecture

What's On For Today?

1. Libraries
2. Concurrent Operations
3. Run Loops & Timers
4. UI Feedback

2.0 Concurrent Operations

- **Last Lecture:** "don't lock up the UI with network requests"
- **This Lecture:** "don't lock up the UI with any long tasks"
- Test on a device to find out what "long" feels like -- do this BEFORE optimizing

2.1 Threading I

- Most apps have a **main thread** that executes code (including responding to the user's input)
- Apps start with a single thread but can create more

2.2 Threading II

- UI events (touches, etc) ☐ **must be processed** on main thread in iOS
- An app will not respond to touches if the main thread is busy
- `NSURLConnection` solved this problem for us, remember?

2.3 iOS Handles Threading For You

- **Avoid** creating threads explicitly
- Use API methods where possible
- Many methods do the hard work for you "automagically":
 - NSURLConnection (getting URL)
 - NSXMLParser (parsing XML)
 - NSOperationQueue
 - ... more

2.4 Threading Example

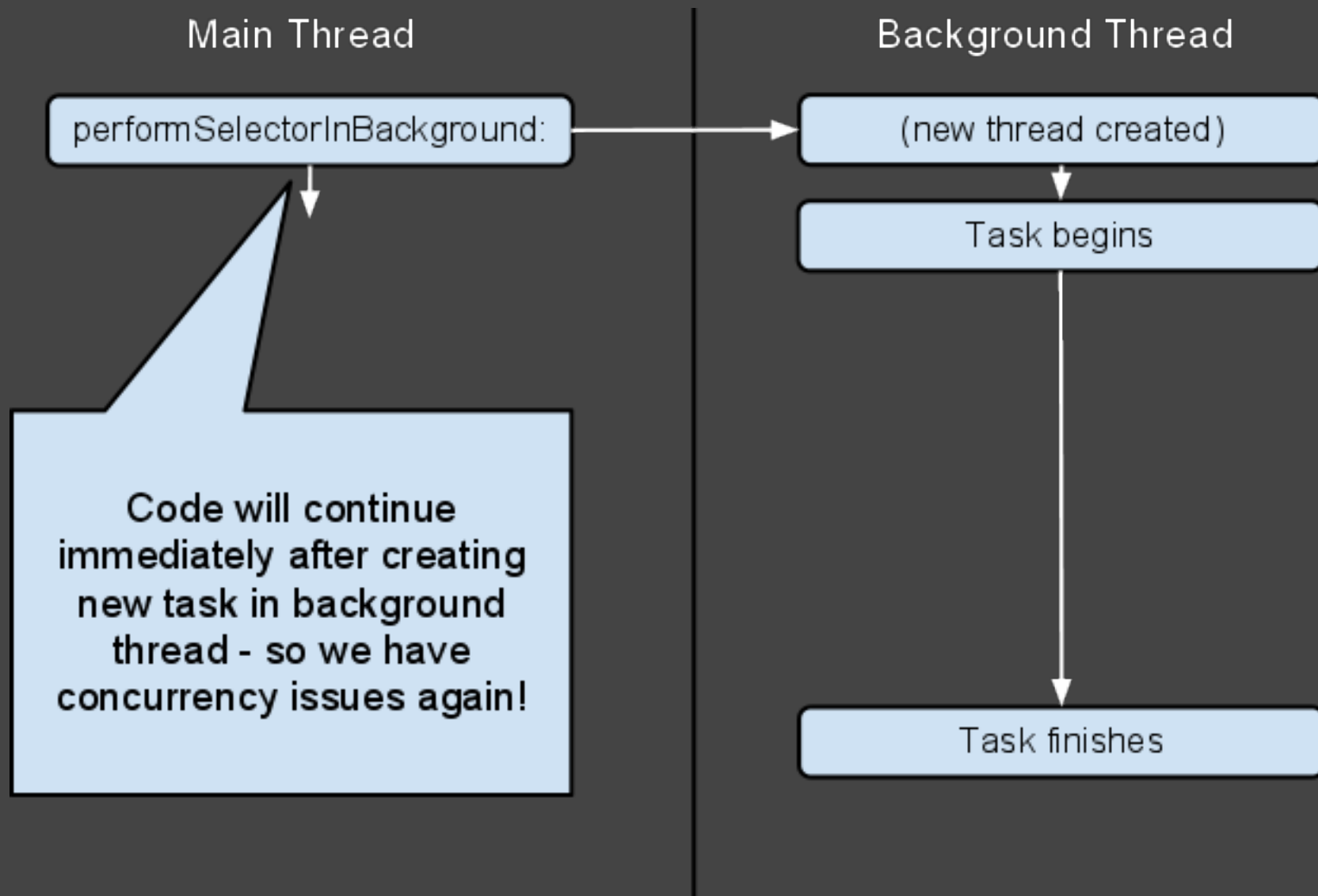
But sometimes we have to go it alone:

```
// Automatically creates background thread and  
// runs code using it
```

```
[self performSelectorInBackground:@selector(parseData:)  
    withObject:hugeData];
```

```
- (void) parseData:(NSData*)hugeData {  
    // You need this code for any background thread!  
    NSAutoreleasePool *p=[[NSAutoreleasePool alloc] init];  
  
    /** Actual parsing code that takes time here **/  
    [p release];  
}
```

2.5 Threading Example Visualized

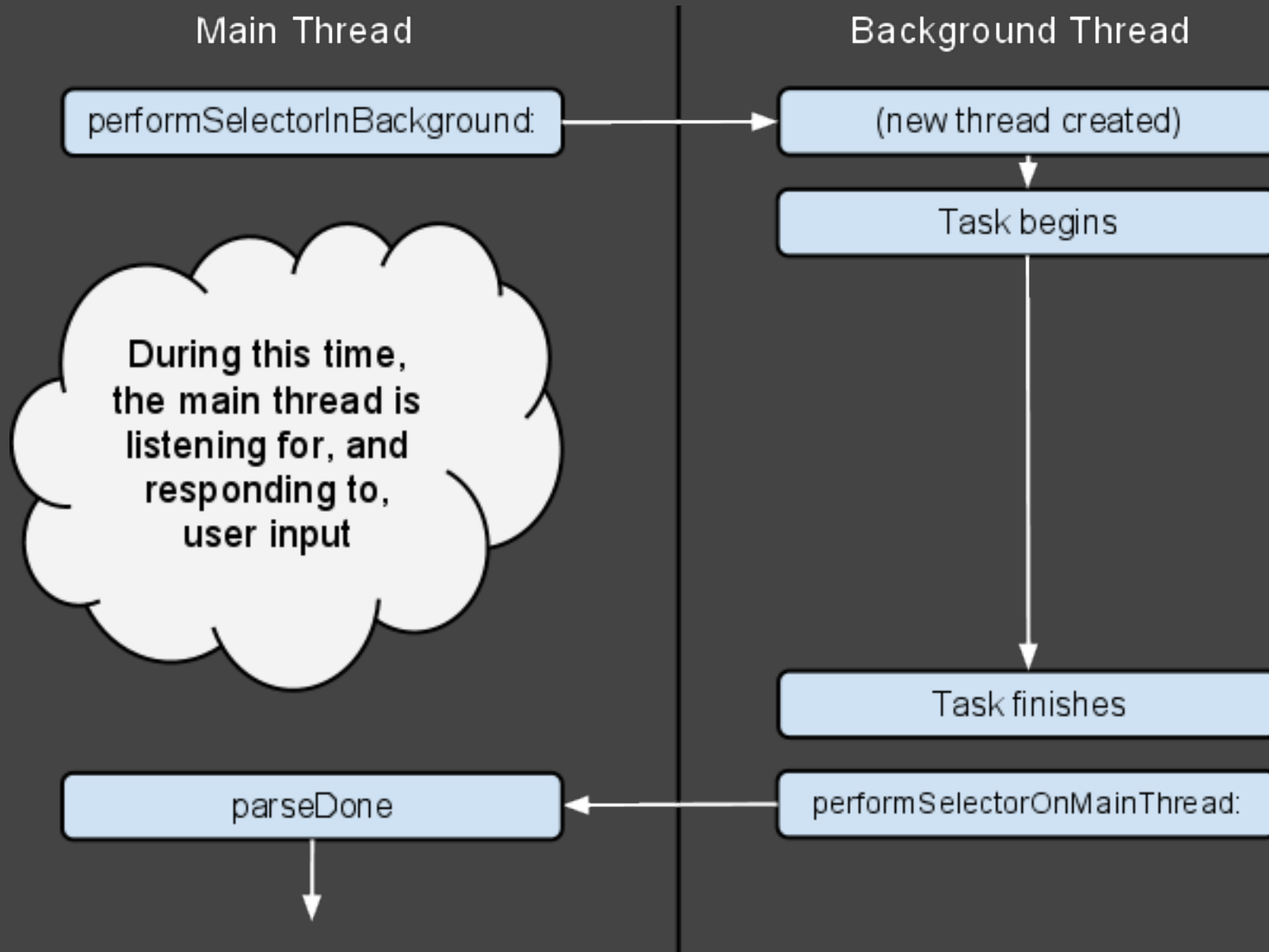


2.6 Threading Example, Improved

Also need to know when finished:

```
- (void) parseXML:(id)xmlTree {  
    NSAutoreleasePool *p=[[NSAutoreleasePool alloc] init];  
  
    /** XML parsing code that takes time here **/  
  
    // Danger! calling [self parseDone] causes problems!!  
    // Must explicitly say we want this on MAIN THREAD  
  
    [self performSelectorOnMainThread:@selector(parseDone)  
        withObject:nil  
        waitUntilDone:NO];  
    [p release];  
}
```


2.7 Threading Example, Visualized Again



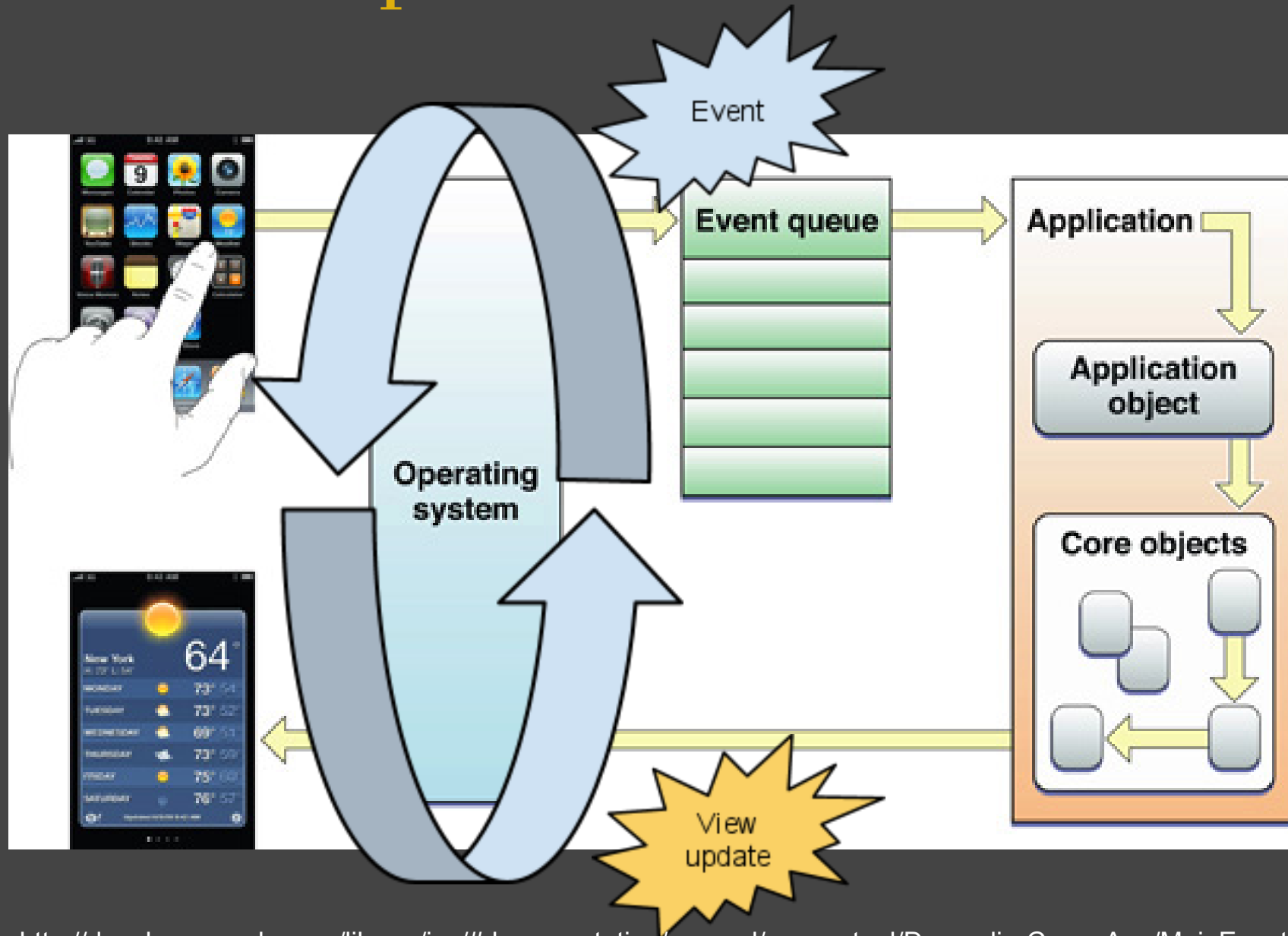
What's On For Today?

1. Libraries
2. Concurrent Operations
3. Run Loops & Timers
4. UI Feedback

3.0 Run Loops

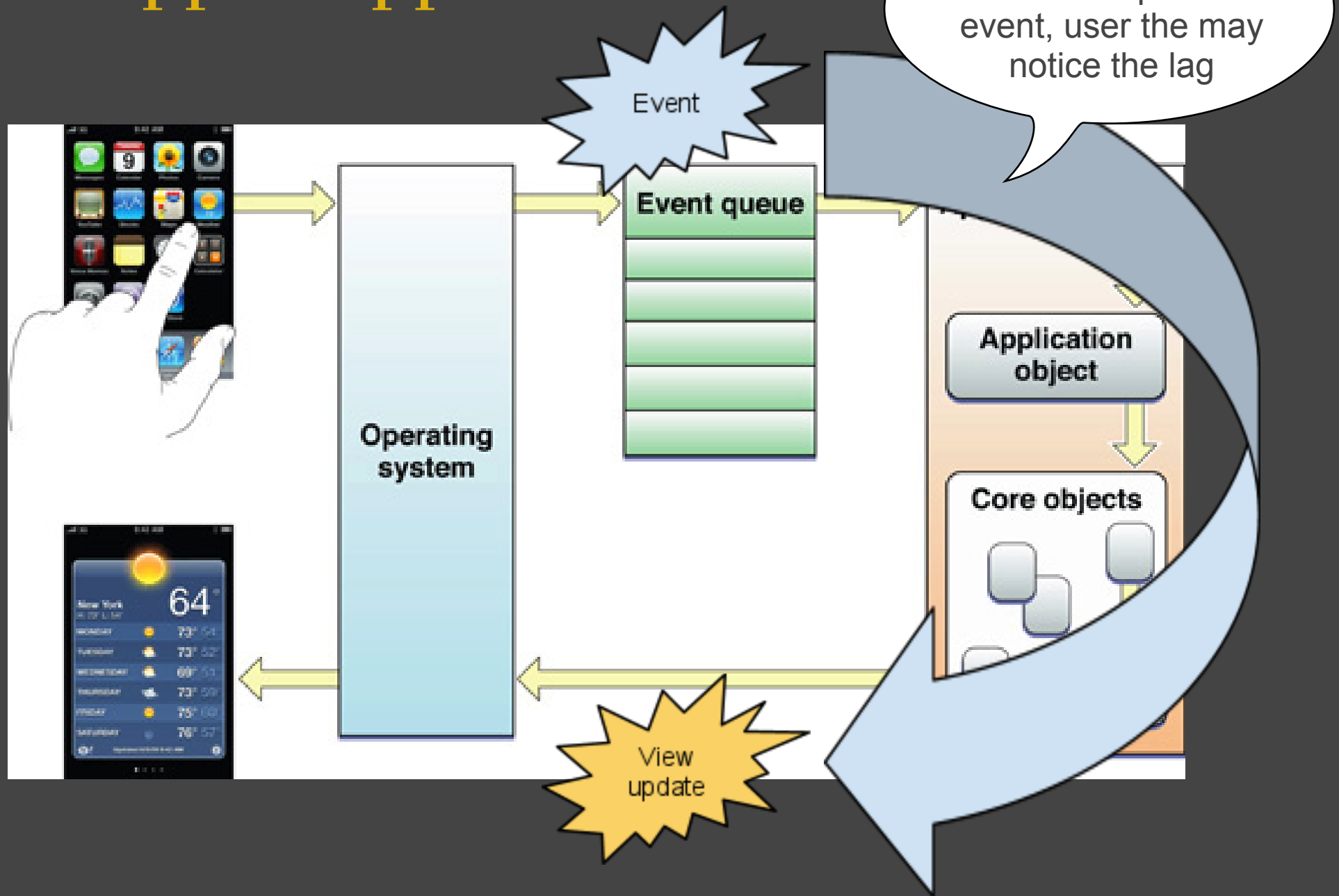
- In iOS, every thread has 1 **run loop** (but they are not the same thing)
- A run loop is exactly what it says: ***code that loops repeatedly***
- Understanding run loops helps understand event-driven apps

3.1 Run Loops Visualized



Source: <http://developer.apple.com/library/ios/#documentation/general/conceptual/Devpedia-CocoaApp/MainEventLoop.html>

3.2 "App Snappiness"



3.3 Run Loops Handle Input Sources

Each cycle, the run loop checks its **input sources**:

- Was the screen touched?
- Is the system turning off?
- Did the phone ring? SMS received?
- Did a timer fire?

If none, it checks a short time later

3.4 Run Loops And You

- Run loops are important to know about if you use timers
- The NSTimer class uses the run loop to call some code at a later interval

Source: http://developer.apple.com/library/ios/#documentation/Cocoa/Reference/Foundation/Classes/nstimer_Class/Reference/NSTimer.html

3.5 NSTimer Example

```
// Assume "@property (retain) NSTimer *timer;" in .h
```

```
// Calls -timerFired: every second from now on
```

```
self.timer = [NSTimer scheduledTimerWithTimeInterval:1.0
                target:self
                selector:@selector(timerFired:)
                userInfo:nil
                repeats:YES];
```

```
- (void) timerFired:(NSTimer*)timer {
    // The code for the iPhone's Clock app is probably
    // a lot like this, updating the value on the screen
    // every second
}
```


3.6 Canceling an NSTimer

Stop/cancel a timer by invalidating it:

```
[self.timer invalidate]; // "you're fired!"  
self.timer = nil;       // Common to nil out afterward
```

Always invalidate a timer when you don't need it anymore. Often done in the class' -dealloc method.

3.7 Don't Say We Didn't Warn You

Failing to invalidate a timer is a leading cause of premature hair loss.



What's On For Today?

1. Libraries
2. Concurrent Operations
3. Run Loops & Timers
4. UI Feedback

4.0 UI Feedback

Last lecture:

- UI feedback tools summary
- Handling failures

This lecture:

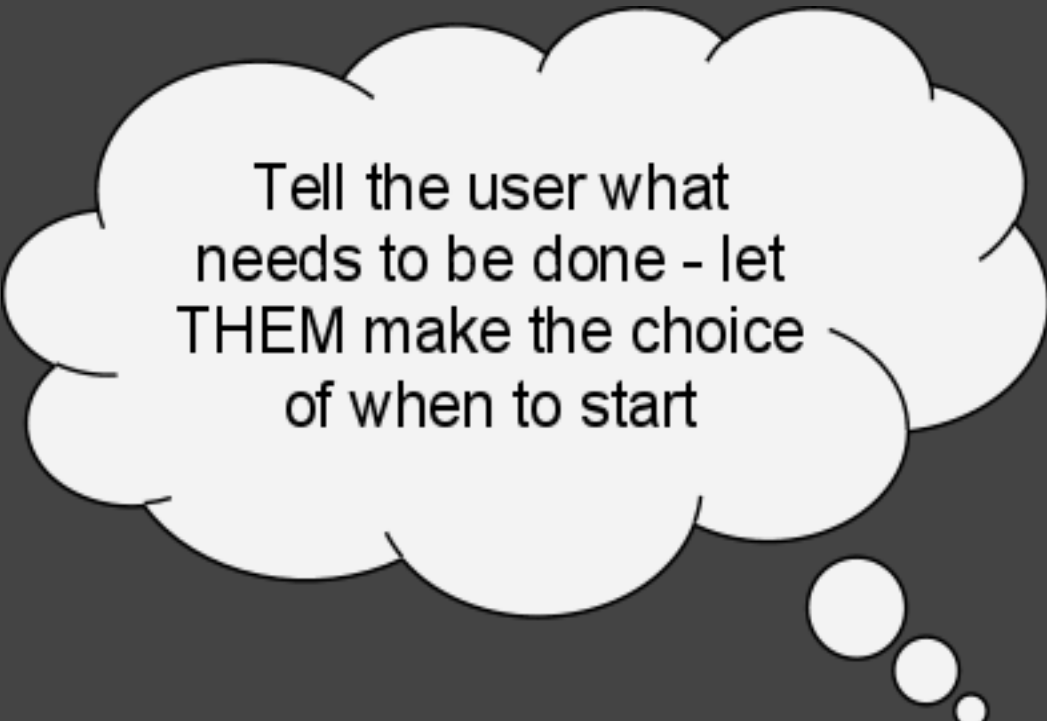
- Keeping the user in control
- UX Flow, UX Design theory

4.1 Keeping the User in Control

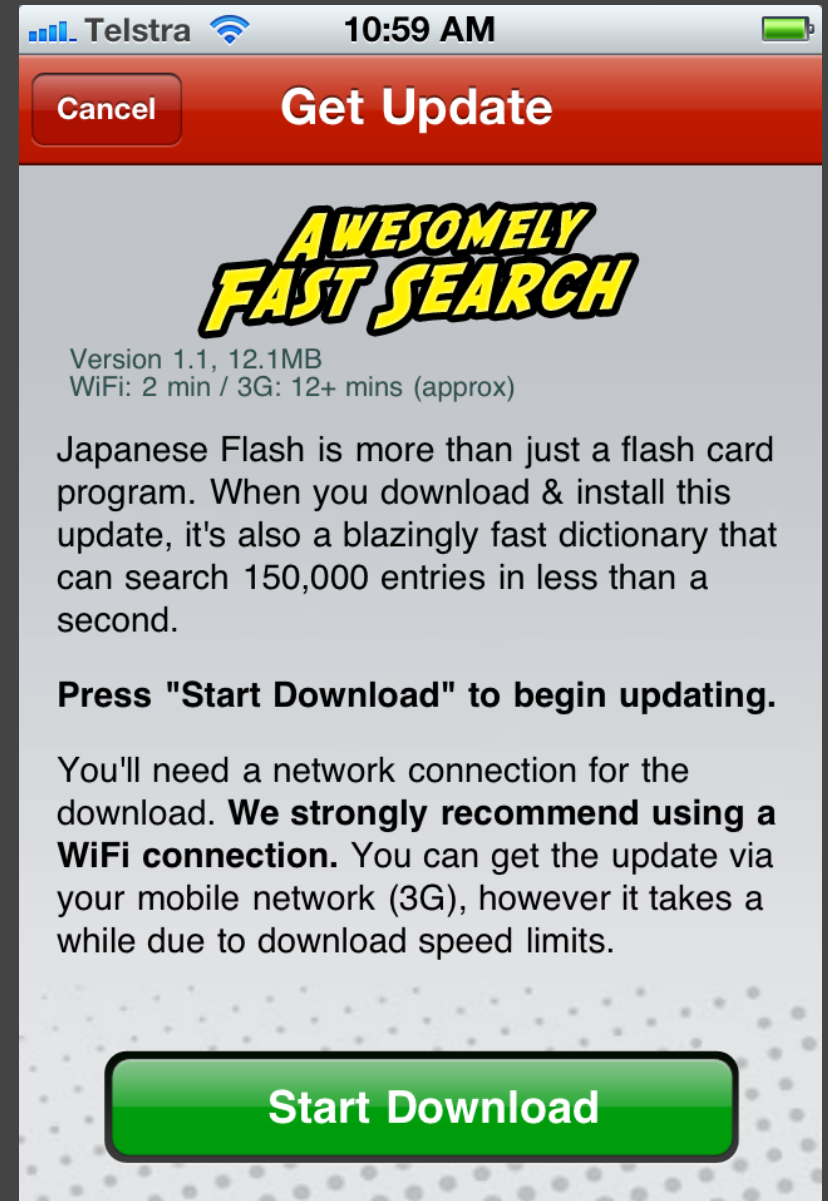
Assume we have a 10~30 sec task.

- Blocking the UI ~ *Unacceptable*
- Modal feedback ~ *Meh*
- Modal feedback w/ cancel ~ *Nice*
- Background execution ~ *Brilliant*□


4.2 Modal Feedback with Cancel/Pause



Tell the user what
needs to be done - let
THEM make the choice
of when to start

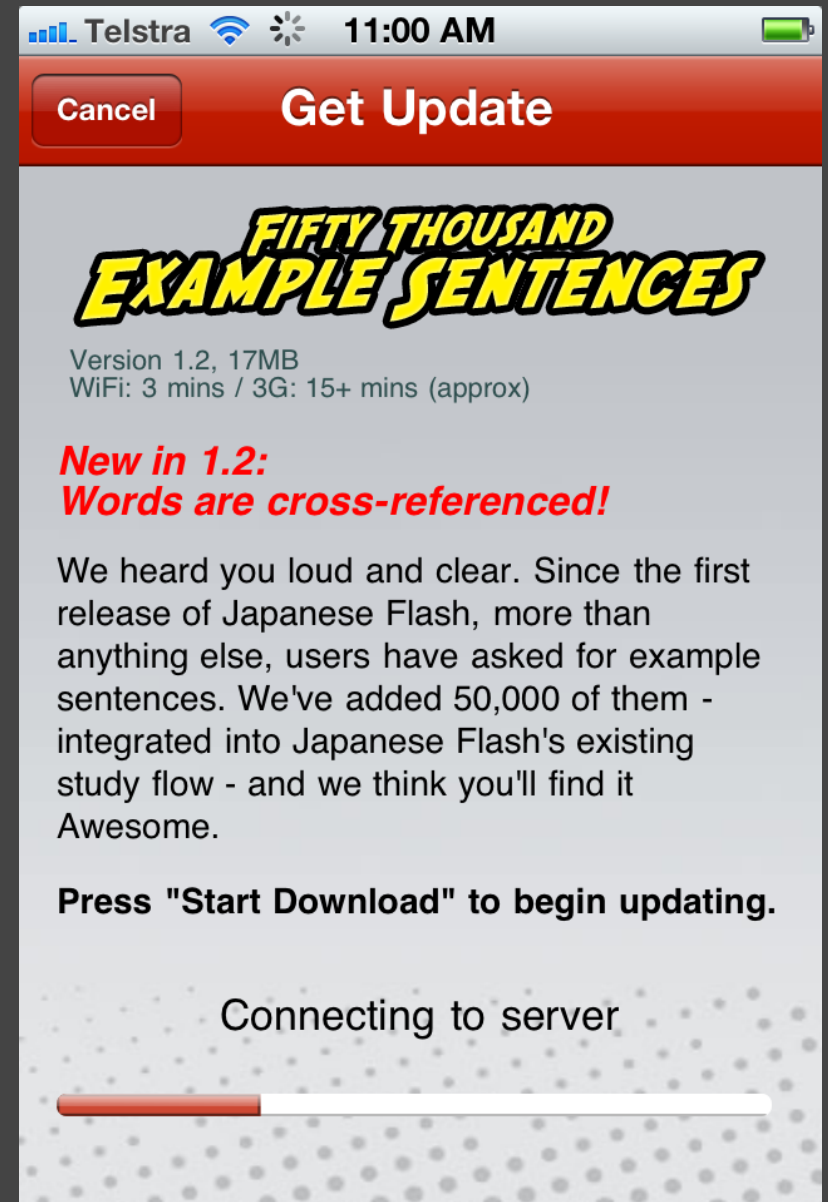


4.3 Modal Feedback with Cancel/Pause



Allow the user to get out, even in the middle of the task - people change their minds!

Most likely, ***Finish Later*** would be better than ***Cancel***, from the user's perspective



4.4 Background Task Completion

If possible, allow user to continue using app while task completes in background

Possibly, the red screen could shrink slightly, and a UIProgressbar at the top could inform the user of the current task's progress?



4.5 How to Keep the User in Control

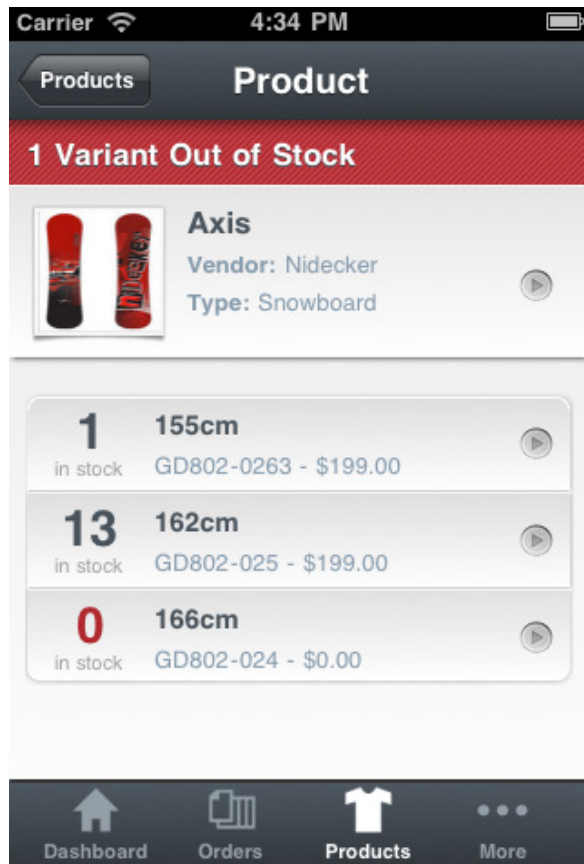
1. Plan UX before you start coding!
2. Use MVC patterns to keep task code loosely coupled to the application's view controllers
3. Allow others to play with your app
(Keeping your hands to yourself, just watch)

4.6 UX Paradigms

- The mobile UX paradigm has been strongly influenced by Apple's iOS
- Through repetition, a UI or UX choice becomes accepted as "standard"
- Taking a fresh approach is great, but avoid straying too far from what users know

4.7 iOS UX Paradigm Examples

- Navigation at the top
- Drill-down content in tables
- Toolbars at the bottom
- Tab bar items for different functions
- Closing an app and re-opening will maintain state



SHOPIFY



TWEET BOT



ANIMAL PHONE

4.7.1 A Consistent UX Paradigm

4.8 Closing Thoughts on UX Design

- Use lots of apps, observe what works and what doesn't
- Form your own opinion
- Try new things, but be brave enough to "kill off" ideas that don't work
- Test on real people
- Keep an open mind

4.9 Learn How To Break The Rules

- Ignoring the paradigm is bad
- First, learn the paradigm intimately
- By understanding the rules of the paradigm we learn how to break them "properly"

What We Covered Today

1. Libraries
2. Concurrent Operations
3. Run Loops & Timers
4. UI Feedback

End of Lecture 7

1. Lab
2. Assignments
3. Be on time