Johns Hopkins Engineering

User Experience

Module 3

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Introduction

Module 2 Recap

- UIViewControllers
- Storyboards and Segues
- Autolayout
- Size Classes

Module 3 Topics

- Threading on iOS
- Asynchronous Multithreading
- Gestures
- Instruments

Reminders

- Homework 1 is due at the end of this module
- Development Log 2 assignment is now available

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Discussion

This week’s discussion topic

- How does Grand Central Dispatch (GCD) differ from other threading mechanisms?
- What other tools have you used for profiling?
  - Compare/contrast with Instruments.

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Threading on iOS

Threading on iOS

- Your app runs on one or more threads
  - The first of which is started by the OS
  - Your app may spawn off other threads to execute specific tasks
- Threads are exposed at various levels:
  - NSThread
  - POSIX Threads

The cost of a thread

- Threads are helpful, but keep in mind their cost:
  - Memory Space
    - Kernel Space (~1KB)
    - Program’s Memory Space (~1-2MB)
  - Creation Time (~90 microseconds)
    - Processor Load
    - Speed of Computer
    - Available Memory

Creating an NSThread

- detachNewThreadSelector:toTarget:withObject (class method)
  - Selector: the thread's entry point
  - Target: the object that defines the method
  - Object: any data you wish to pass into the thread
- Create a new NSThread instance
  - Same parameters as above
  - Don't forget to start the thread!

Creating POSIX Threads

- macOS and iOS both support C-based POSIX Threads
- Allows for cross platform support
- Keep in mind that NSThread is the main interface for making threads in Cocoa and Cocoa Touch apps, you can use POSIX threads if they are more convenient

https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/Multithreading/CreatingThreads/CreatingThreads.html

Performing a Selector on a Thread

- If you wish to dispatch the execution of a method to a thread, there is built in support, courtesy of a method in NSObject:
  - `performSelector(onMainThread:with:waitUntilDone:)` performs the selector on the main thread
  - `perform(_:on:with:waitUntilDone:)` performs the selector on the specified thread (_)
  - `performSelector(inBackground:with:)` performs the selector on a new background thread

Terminating a Thread

- While you should try to let your thread run to completion, sometimes you can’t – for instance, when a user cancels an operation
- Be sure to design your thread to handle a cancel or exit message
- If you do have to cancel, be sure to clean up and exit gracefully!

https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/Multithreading/CreatingThreads/CreatingThreads.html

JOHNS HOPKINS

WHITING SCHOOL
of ENGINEERING