**Multiple Screens without a ~~Navigation Controller~~ Segue**Note: There are slides for this topic!

Intro

* See how everyone is doing with size classes- look at someone’s code?
* Look at due dates

Overview

* Multiple content views require multiple view controllers
* Switching between views can be done with a Navigation Controller (multiple types) and segues, or by your own code.
* The textbook (Ch. 6, *Beginning iPhone Development with Swift*) example uses your own code.
* The Xamarin guide uses a Navigation Controller either with or without a segue.

Switching Views with Your Own Code

* Textbook example walk-through (yellowView, blueView)
* Steps for implementing your own view switching code:
* Create the main view – test it (don’t do this in the order the textbook does it)
* Create the second view, add a button to the main view that launches the second view – test it.
* Add a button to the second view that takes you back to the main view –test it.

Frame

This rectangle defines the size and position of the view in its superview’s coordinate system. You use this rectangle during layout operations to size and position the view. Setting this property changes the point specified by the [center](https://developer.apple.com/reference/uikit/uiview/1622627-center) property and the size in the [bounds](https://developer.apple.com/reference/uikit/uiview/1622580-bounds) rectangle accordingly. The coordinates of the frame rectangle are always specified in points.

Changing the frame rectangle automatically redisplays the receiver without invoking the [draw(\_:)](https://developer.apple.com/reference/uikit/uiview/1622529-draw) method. If you want the [draw(\_:)](https://developer.apple.com/reference/uikit/uiview/1622529-draw) method invoked when the frame rectangle changes, set the [contentMode](https://developer.apple.com/reference/uikit/uiview/1622619-contentmode) property to [redraw](https://developer.apple.com/reference/uikit/uiviewcontentmode/1622652-redraw).

<https://developer.apple.com/reference/uikit/uiview/1622621-frame>

Lazy Loading

Tip Lazy loading is a key component of resource management on iOS, which you should implement

anywhere you can. In a complex, multi-view application, being responsible and flushing unused objects from

memory can be the difference between an application that works well and one that crashes periodically

because it runs out of memory.

Container View Controller

This code makes the incoming view controller a child of the switching view controller. View controllers

like SwitchingViewController that manage other view controllers are referred to as container view

controllers . The standard classes UITabBarController and UINavigationController are both container

view controllers and they have code that does something similar to what the switchViewController(from:,

to:) method is doing.

About View Hierarchies

The root controller controls a content view. That view can have sub-views. In the Ch. 6 example the, the root view is the one in the SwitchingViewController. It never gets switched out, the Blue and Yellow views just get loaded (and unloaded) as child views.

*// Reference: iOS Views - Apple programing guide for iOS*

*//*[*https://developer.apple.com/library/ios/documentation/WindowsViews/Conceptual/ViewPG\_iPhoneOS/CreatingViews/CreatingViews.html#//apple\_ref/doc/uid/TP40009503-CH5-SW47*](https://developer.apple.com/library/ios/documentation/WindowsViews/Conceptual/ViewPG_iPhoneOS/CreatingViews/CreatingViews.html#//apple_ref/doc/uid/TP40009503-CH5-SW47)

*// Espeically look at "Creating and Managing a View Hierarchy"*