CSC 581: Mobile App Development

Spring 2018

Unit 3: Navigation design

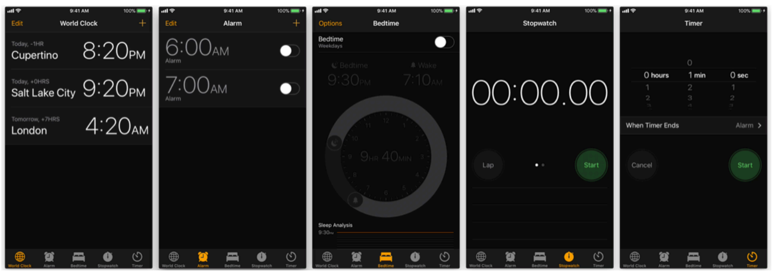
- UIKit
  - tab bar controllers
  - view controller life cycle
- navigation hierarchies
- design guidelines

1

Tab bar controllers

a tab bar allows you to arrange your app into distinct sections

- e.g., Apple Clock app: World Clock, Alarm, Bedtime, Stopwatch, Timer



a tab bar is defined using a Tab Bar View

- shows tabs (icons) across the bottom of the screen
- each tab has its own navigation hierarchy
- the tab bar controller coordinates the navigation

2

Adding a tab bar

first, create the main View for your project

- select that View in IB, then choose Editor > Embed in > Tab Bar Controller
- this adds a Tab Bar Controller to the canvas as well as adding a UITabBarItem to the bottom of the View

to add another tab bar item

- drag a View Controller from the Object Library to the canvas
- connect it by control-dragging from the Tab Bar Controller to the new View Controller, and choosing "view controllers" under Relationship Segue
- this adds a second tab bar item to the Tab Bar Controller

customize the tab bar item

- select the desired tab bar item in a View
- in the Attributes Inspector, select the desired icon under System Item
  e.g., Favorites, Features, Search, Downloads, …
- you can change the title under Bar Item > Title
- you can select a custom icon image under Bar Item > Image

3

Adding behavior to the Views

by default, the first View has a ViewController.swift file already defined

- as we have done before, can add UI elements to the View and link to Outlets and Actions in the ViewController

to add ViewControllers for the new View

- select File > New > File… > Cocoa Touch Class
- set the subclass of the new class to UIViewController and give it a unique name

- connect the View and the new ViewController by selecting the ViewController in the Document Outline (left side of IB)
- then, in the Identity Inspector (icon to left of Attributes Inspector), set the custom class to the new Swift file

HW3b: complete the guided project (Personality Quiz) at the end of Unit 3

- substitute your own questions and quiz results

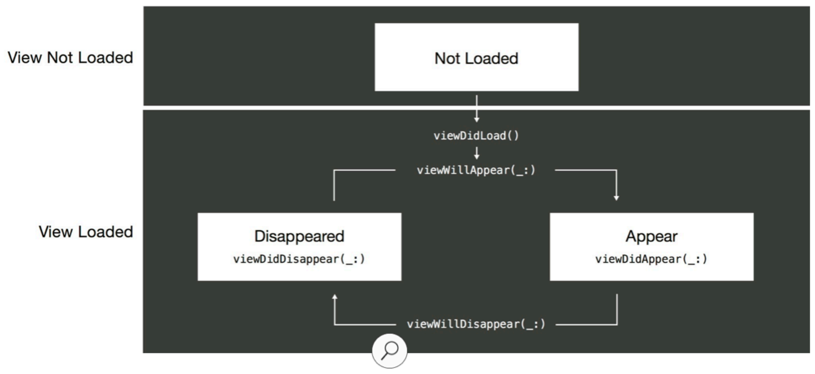
4

View Controller life cycle

view controllers can be in one of 5 states

1. not loaded
2. appearing
3. appeared
4. disappearing
5. disappeared

as a view transitions from one state to another, iOS calls a built-in method



5

View event management

viewDidLoad()

- performs tasks that need to be doneonce, when app is first loaded
  e.g, update a label's font, size, or color

viewWillAppear()

- performs tasks that need to be done each time the view is to appear on the screen
  e.g, refreshing views, adjusting to new orientation, accessing location

viewDidAppear()

- waits until the view is fully loaded – better for complex or slow tasks
  e.g., starting an animation, fetching data

viewWillDisappear()

- performs tasks that need to be done when the user navigates away from the screen
  e.g, refreshing views, adjusting to new orientation, accessing location

viewDidDisappear()

- waits until the user has navigated to a new view
  e.g., stop services related to the old view (such as background audio)

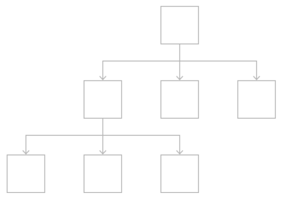
6

Navigation hierarchies

hierarchical

- user makes one choice per day until reaching a destination

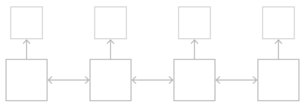
  e.g., Settings, Mail



flat

- user switches between multiple content categories (e.g., with tab bar)

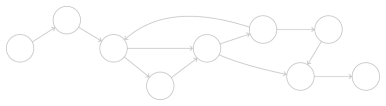
  e.g., Music, App Store



content- or experience-driven

- the content defines the navigation

  e.g., games, books, other immersive apps



7

Navigation design guidelines

Design an information structure that makes it fast and easy to get to content.

- Organize your information in a way that requires a minimum number of taps, swipes, and screens.

Use standard navigation components.

- Whenever possible, use standard navigation controls, such as tab bars, segmented controls, table views, collection views, and split views. Users are already familiar with these controls and will intuitively know how to get around in your app.

Use a navigation bar to traverse a hierarchy of data.

- The navigation bar's title can display the user's current position in the hierarchy, and the Back button makes it easy to return to the previous position.

Use a tab bar to present peer categories of content or functionality.

- A tab bar lets people quickly and easily switch between categories or modes of operation, regardless of their current location.

Use the proper transition style.

- Whenever you're transitioning to a new screen to display more detail about an item, you should use a right-to-left push transition used by a Show segue within a navigation controller. If the user is switching contexts - from displaying contacts to adding a new contact, for example - use a modal transition.

8