

CSC 471 / 371
Mobile Application
Development for iOS



Prof. Xiaoping Jia
School of Computing, CDM
DePaul University
xjia@cdm.depaul.edu
[@DePaulSWEEng](https://twitter.com/DePaulSWEEng)

1

Adaptive Design

2

Outline

- Size classes
- Split view controller



DEPAUL UNIVERSITY

3

Size Classes

4

Size Classes

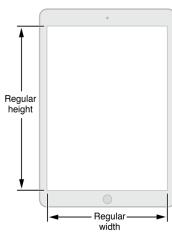
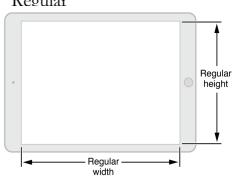



- Introduced in iOS 8
- A *size class* identifies a relative amount of display space for the height and for the width.
- Two size classes: *regular* and *compact*.
 - The *regular* size class – expansive space
 - The *compact* size class – constrained space.
- Each display environment is characterized by a *horizontal* size class and a *vertical* size class.
- iOS automatically makes various layout changes when the size classes of a display environment change.

DEPAUL UNIVERSITY

5

The Size Classes of iPad

- Portrait: Regular x Regular
- Landscape: Regular x Regular

DEPAUL UNIVERSITY

6

The Size Classes of an iPhone 6+

- Portrait: Compact x Regular
- Landscape: Regular x Compact

The diagram illustrates the size classes for an iPhone 6+. In portrait mode, the device is labeled with 'Regular height' and 'Compact width'. In landscape mode, it is labeled with 'Compact height' and 'Regular width'. A callout box states: 'The portrait and landscape modes of the same screen are two distinct display environments'.

DEPAUL UNIVERSITY 7

The Size Classes of iPhone 6

- Portrait: Compact x Regular
- Landscape: Compact x Compact

The diagram illustrates the size classes for an iPhone 6. In portrait mode, the device is labeled with 'Regular height' and 'Compact width'. In landscape mode, it is labeled with 'Compact height' and 'Compact width'. Callout boxes state: 'The portrait mode of iPhone 6/6s and 6/6s+ have the same size classes' and 'The landscape mode of iPhone 6/6s and 6/6s+ have different size classes'.

DEPAUL UNIVERSITY 8

Adaptive Design with Size Classes

- You can design adaptive UI based on the size classes
- Xcode allows selection of size classes for the width and height from
 - Compact, Any, Regular
- Widgets and constraints can be *installed* or *uninstalled* for specific size classes
 - When a widget or constraint is *uninstalled* the associated object *does not exist* in the view hierarchy

DEPAUL UNIVERSITY 9

The Default Size Class

- The default size class of the Interface Builder is Any Width | Any Height
 - All widgets are installed for screens of all size classes
 - The same set of constraints are installed, i.e., in effect, for screens of all size classes

A screenshot of the Xcode Interface Builder showing the size class selector. It displays 'Any' selected under both width and height categories. A callout box points to the 'ANY' label.

DEPAUL UNIVERSITY 10

Select Size Classes

- Use the size class control
- Move the mouse to select from the 3×3 grid
- The square shape of the scene represents the default size class w Any | h Any

A screenshot of the Xcode Interface Builder showing the size class control. A callout box points to the 'Size class grid' which shows a 3×3 grid of size classes. Another callout box points to the 'Size class control' at the bottom left. A third callout box points to the 'w Any | h Any' label at the bottom right.

DEPAUL UNIVERSITY 11

The Adaptive View Demo

- Different layouts in *Portrait* and *Landscape* modes
 - Different set of constraints for different size classes
 - Some widgets in specific size class

Two screenshots of an adaptive view demo. The left screenshot shows the view in 'Portrait Mode' with three buttons labeled 'One', 'Two', and 'Three' in yellow, cyan, and green respectively. The right screenshot shows the view in 'Landscape Mode' where the buttons are rearranged into two rows: 'One' and 'Two' in the top row, and 'Three' in the bottom row.

DEPAUL UNIVERSITY 12

The Adaptive View

- Start with a single view app
- Use the Default size class
- Design the UI for the default size class first
 - The Portrait mode
 - Customize for the landscape size class later

DEPAUL UNIVERSITY 13

The Portrait Model UI Design

- The constraints:
 - Center X Alignment:
 - View – One
 - View – Two
 - View – Three
 - Center Y Alignment:
 - View – Two
 - Vertical Space:
 - One – Two
 - Two – Three

DEPAUL UNIVERSITY 14

Run ...

- Same layout in *Portrait* and in *Landscape*
- Not adaptive
- Not suited for the *Landscape* mode

DEPAUL UNIVERSITY 15

Change the Size Class

- Use the Size Class control to set the size class to w Any | h Compact
- The size class that fits the *Landscape* mode of all iPhones, including iPhone 6+

DEPAUL UNIVERSITY 16

Uninstall Constraints

- Keep only the constraints for *Two*. Uninstall the constraints for *One* and *Three*
- Select the constraints to be uninstalled
- In the *Constraint Inspector*
 - Click the '+' button

DEPAUL UNIVERSITY 17

Uninstall Constraints

- Add a size class Any Width | Compact Height
- Uncheck the *installed* box for W Any H C
- Now, all the selected constraints are uninstalled for the size class Any Width | Compact Height

DEPAUL UNIVERSITY 18

Install New Constraints

- Add constraints
 - Center Y Alignment
 - View – One
 - View – Three
 - Horizontal Space
 - One – Two
 - Two – Three
- These constraints will only be installed for Any | Compact

DEPAUL UNIVERSITY 19

Run ...

- The Landscape model looks better

DEPAUL UNIVERSITY 20

Add a Widget

- Add a Label and constraints
 - Center X Alignment
 - Vertical Space to top

DEPAUL UNIVERSITY 21

Run ...

The label appears only in the *Landscape* mode

DEPAUL UNIVERSITY 22

A More Complex Adaptive Design

- The Controls app adapted to support Portrait and Landscape layout
 - Same widgets. Different constraints
 - No change in the view controller

DEPAUL UNIVERSITY 23

A More Complex Adaptive Design

- Divide the view into several regions, i.e., containers
- Each container maintain its own set of constraints.
- Only the top level constraints need to be adapted

DEPAUL UNIVERSITY 24

Split View Controller



25

Split View Controller



26

- Designed for iPad. Works on all iOS 8 devices
 - iPad only for iOS 7 or earlier
- Manages two child view controllers – master/detail.
 - Several presentation styles based on screen dimensions
 - Side-by-side, one at a time, partial overlay

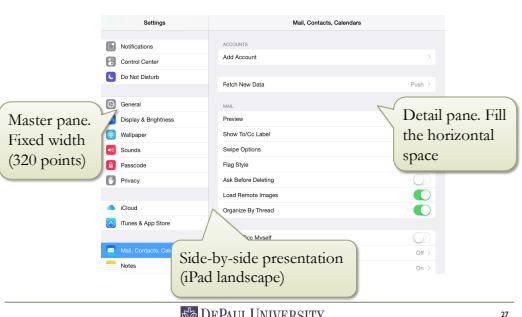


DEPAUL UNIVERSITY

Split View Controller Example



27



Master pane. Fixed width (320 points)

Detail pane. Fill the horizontal space

Side-by-side presentation (iPad landscape)

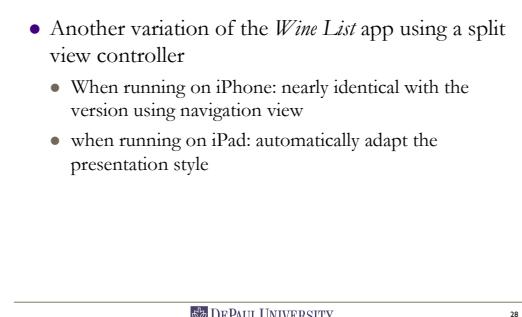
DEPAUL UNIVERSITY

Wine List – Split View



28

- Another variation of the *Wine List* app using a split view controller
 - When running on iPhone: nearly identical with the version using navigation view
 - when running on iPad: automatically adapt the presentation style



The initial view. Tap the Back navigation button to reveal the master view.

The detail view. Always present

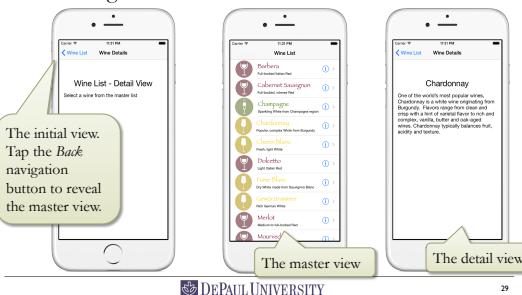
DEPAUL UNIVERSITY

Wine List – Split View



29

- Running on iPhone – one view controller at a time



The initial view. Tap the Back navigation button to reveal the master view.

The master view

The detail view

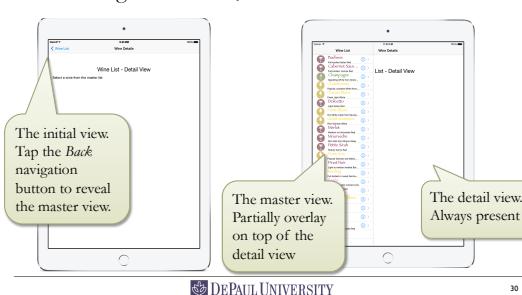
DEPAUL UNIVERSITY

Wine List – Split View



30

- Running on iPad Air, *Portrait* mode.



The initial view. Tap the Back navigation button to reveal the master view.

The master view. Partially overlay on top of the detail view

The detail view. Always present

DEPAUL UNIVERSITY

Wine List – Split View

- Running on iPad Air, *Portrait* mode.

DEPAUL UNIVERSITY 31

Wine List – Split View

- Running on iPad Air, *Landscape* mode.

DEPAUL UNIVERSITY 32

Create a Split View App

- Start with a new single view app
- In storyboard, add a *Split View Controller* (a 4-controller bundle)
 - and move the start arrow to the *Split View Controller*
- The *Split View Controller* has two sub-view controllers
 - The *Master* view controller, which is a *Navigation Controller* with its root view controller
 - The *Detail* view controller

DEPAUL UNIVERSITY 33

The Split View Controller – The Initial Storyboard

The Master view controller, a *Navigation Controller*
The Root view controller of the *Navigation controller*
The relationship segue – *Master* view controller.
The relationship segue – *Root* view controller.
The *Detail* view controller
DEPAUL UNIVERSITY 34

Replace the Detail View Controller

- The initial detail view controller is a simple view controller.
 - No navigation bar when running in iPad *Portrait* mode.
 - Use a *right-swipe* gesture to reveal the master list. No visual clue.
- Replace it with a *Navigation Controller*
 - Has a navigation bar, and *Back* button to reveal the master list.

DEPAUL UNIVERSITY 35

Replace the Detail View Controller

- In storyboard, add a *Navigation Controller*
- Add a segue (Ctrl-Drag) from the *Split View Controller* to the *Navigation Controller*
 - Select: *Relationship Segue | detail view controller*

The Master list
The Detail view
DEPAUL UNIVERSITY 36

Edit the Wine List Scenes

- Nearly the same design and logic from the *Wine List – Navigation+Selection* app
- The *Wine List* and *Wine Detail* scenes and view controllers
- Different** segues from the cells in the *Wine List* scene to the *Navigation Controller*
 - Choose: *Selection Segue* | *Show detail* (instead of *Show*)

DEPAUL UNIVERSITY 37

The Completed Storyboard

The storyboard diagram illustrates the completed adaptive design. It shows a **Split View Controller** connected to a **Navigation Controller**. The **Navigation Controller** manages two scenes: **The Wine List** (a **Table View**) and **The Wine Detail** (a **Wine List - Detail View**). Segues connect cells in the **Wine List** table to the **Navigation Controller**, specifically using the *Show detail* segue type. Callouts point to these elements with labels: "The Wine List scene", "The Show detail segues", and "The Wine Detail view".

DEPAUL UNIVERSITY 38

The Wine Detail View Controller

- Only change needed in the *Detail View Controller*
 - To set up the *Back* navigation button

```
override func viewDidAppear(animated: Bool) {
    navigationItem.leftBarButtonItem =
        splitViewController?.displayModeButtonItem()
    navigationItem.leftItemsSupplementBackButton = true
    if let w = wine {
        titleLabel.text = w.name
        descriptionLabel.text = w.longDescription
    }
}
```

DEPAUL UNIVERSITY 39

The Wine List View Controller

- Only change needed in the *Wine List View Controller*
 - To display the alert as a *Popover*

```
override func tableView(tableView: UITableView,
accessoryButtonTappedForRowWithIndexPath indexPath: NSIndexPath) {
    let alertController = UIAlertController(...)
    let popover = alertController.popoverPresentationController
    if let tableView = self.view as? UITableView {
        popover?.sourceView =
            tableView.cellForRowAtIndex(indexPath)
    }
    presentViewController(alertController, animated: true,
completion: nil)
self.tableView.deselectRowAtIndexPath(indexPath)
}
```

DEPAUL UNIVERSITY

Action Sheet as a Popover

The two screenshots show the *Cover Selection* action sheet displayed as a popover on an iPhone. The action sheet lists various options for selecting a cover, such as "Front Cover", "Back Cover", "Left Cover", and "Right Cover". Each option has a small icon next to it. The popover is anchored to the bottom of the screen.

DEPAUL UNIVERSITY 41

Sample Code

- Adaptive View.zip
- Controls – Rotate.zip
- Wine List – Split View.zip

DEPAUL UNIVERSITY 42

Next ...



- Research project/thesis
- Independent studies
- *Advanced Mobile Application Development*
 - Stay tuned for AY 2016-17

❖ iOS is a trademark of Apple Inc.

DEPAUL UNIVERSITY