Announcements

• Lab 3 is due on Wednesday Sept 30th

• Lab 4 is posted
  – Due Oct 14th

Topics

• NSUserDefaults and Screen Rotation

• View Controllers

• Application Data Flow

• Customizing Navigation

• Tab Bar Controllers

• Combining Approaches
Loading and Saving Data

- Lots of options out there, depends on what you need
  - NSUserDefaults
  - Property lists
  - SQLite
  - Web services
- Covering in greater depth in a few weeks

Saving State Across App Launches

- NSUserDefaults to read and write prefs & state

  Singleton object:
  + (NSUserDefaults *)standardUserDefaults;

  Methods for storing & fetching common types:
  -(NSInteger)integerForKey:(NSString *)key;
  -(void)setInteger:(int)value forKey:(NSString *)key;
  -(id)objectForKey:(NSString *)key;
  -(void)setObject:(int)value forKey:(NSString *)key;

- Find an appropriate time to store and restore your state
More View Controller Hooks

• Automatically rotating your user interface

• Low memory warnings

Supporting Interface Rotation

-(BOOL)shouldAutorotateToInterfaceOrientation:
    (UIInterfaceOrientation)interfaceOrientation
{
    // This view controller only supports portraits
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}
Supporting Interface Rotation

-(BOOL)shouldAutorotateToInterfaceOrientation:
  (UIInterfaceOrientation)interfaceOrientation
{

  // This view controller supports all orientations
  // except for upside-down.
  return (interfaceOrientation != UIInterfaceOrientationPortraitUpsideDown);
}

Autoresizing Your Views

view.autoresizingMask = UIViewAutoresizingFlexibleWidth |
UIViewAutoresizingFlexibleHeight;

view.autoresizingMask = UIViewAutoresizingFlexibleWidth |
UIViewAutoresizingFlexibleTopMargin;
View Controllers

UIViewController
• Basic building block
• Manages a screenful of content
• Subclass to add your application logic
“Your” and Apple View Controllers

- Create your own UIViewController subclass for each screenful
- Plug them together using existing composite view controllers

“Your” and “Our” View Controllers

- Create your own UIViewController subclass for each screenful
- Plug them together using existing composite view controllers
Navigation Controllers

- Stack of view controllers
- Navigation bar
How It Fits Together

- Top view controller’s view
- Top view controller’s title
- Previous view controller’s title

Modifying the Navigation Stack

- Push to add a view controller (iOS 8 and later)
  -(void)showViewController:(UIViewController *)viewController
      sender:(id);

- Push to add a view controller (iOS 7 and earlier, deprecated in iOS 8)
  -(void)pushViewController:(UIViewController *)viewController
      animated:(BOOL)animated;

- Pop to remove a view controller
  -(UIViewController *)popViewControllerAnimated:(BOOL)animated;
Pushing Your First View Controller

- (void)applicationDidFinishLaunching {
  // Create a navigation controller
  navCtrl = [[UINavigationController alloc] init];

  // Push the first view controller on the stack
  [ navCtrl pushViewController: firstViewController animated: NO];

  // Add the navigation controller’s view to the window
  [window addSubview: navCtrl.view];
}

In Response to User Actions

- Push from within a view controller on the stack

  - (void) someAction: (id) sender
  {
    // Potentially create another view controller
    UIViewController * viewController = ...;

    [ self. navigationController pushViewController: viewController animated: YES ];
  
  }

- Almost never call pop directly!
  - Automatically invoked by the back button
Application Data Flow

A Controller for Each Screen

List Controller → Detail Controller
Connecting View Controllers

- Multiple view controllers may need to share data
- One may need to know about what another is doing
  - Watch for added, removed or edited data
  - Other interesting events
How Not To Share Data

- Global variables or singletons
  - This includes your application delegate!
- Direct dependencies make your code less reusable
  - And more difficult to debug & test

Best Practices for Data Flow

- Figure out exactly what needs to be communicated
- Define input parameters for your view controller
- For communicating back up the hierarchy, use loose coupling
  - Define a generic interface for observers (like delegation)
Customizing Navigation

- Buttons or custom controls
- Interact with the entire screen
**UINavigationItem**

- **Describes appearance of the navigation bar**
  - Title string or custom title view
  - Left & right bar buttons
  - More properties defined in UINavigationBar.h
- **Every view controller has a navigation item for customizing**
  - Displayed when view controller is on top of the stack
Displaying a Title

- **UIViewController already has a title property**
  - @property(nonatomic,copy) NSString *title;
- **Navigation item inherits automatically**
  - Previous view controller’s title is displayed in back button

```swift
viewController.title = @“Detail”;
```

---

Left & Right Buttons

- **UIBarButton Item**
  - Special object, defines appearance & behavior for items in navigation bars and toolbars

- **Display a string, image or predefine system item**

- **Target + action (like a regular button)**
**Text Bar Button Item**

```objective-c
- (void)viewDidLoad
{
    UIBarButtonItem *fooButtonItem = [[UIBarButtonItem alloc]
        initWithTitle:@"Foo"
        style:UIBarButtonItemStyleBordered
        target:self
        action:@selector(foo:)];

    self.navigationItem.leftBarButtonItem = fooButtonItem;

    [fooButtonItem release];
}
```

**System Bar Button Item**

```objective-c
(void)viewDidLoad
{
    UIBarButtonItem *addButtonItem = [[UIBarButtonItem alloc]
        initWithBarButtonSystemItem:UIBarButtonSystemItemAdd
        style:UIBarButtonItemStyleBordered
        target:self
        action:@selector(add:)];

    self.navigationItem.rightBarButtonItem = addButtonItem;

    [addButtonItem release];
}
```
Navigation Controller with Custom Button Demo

Edit/Done Button

- Very common pattern
- Every view controller has one available
  - Target/action already set up

self.navigationItem.leftBarButtonItem = self.editButtonItem;

// Called when the user toggles the edit/done button
(void)setEditing:(BOOL)editing animated:(BOOL)animated
{
  // Update appearance of views
}
Custom Title View

- Arbitrary view in place of the title

UI Segmented Control *segmentedControl = ... self.navigationItem.titleView = segmentedControl;
[segmentedControl release];

Back Button

- Sometimes a shorter back button is needed

Self.title = @"Hello there,";
UI Bar Button Item *heyButton = [[UI Bar Button Item alloc]
initWithTitle:@"Hey!" ...];
self.navigationItem.backBarButtonItem = heyButton;
[heyButton release];
Tab Bar Controllers

UITabBarController

Array of view controllers
How it Fits Together

- Selected view controller’s view
- All view controller’s titles

Setting Up a Tab Bar Controller

```swift
(void)applicationDidFinishLaunching
// Create a tab bar controller
tabBarController = [[UITabBarController alloc] init];

// Set the array of view controllers
tabBarController.viewControllers = [NSArray arrayWithObjects:
    firstVC, secondVC, nil];

// Add the tab bar controller’s view to the window
[window addSubview:tabBarController.view];
```
Tab Bar Appearance

- View controllers can define their appearance in the tab bar
- **UITabBarItem**
  - Title + image or system item
- Each view controller comes with a tab bar item for customizing

Creating Tab Bar Items

- Title and image

```objective-c
- (void)viewDidLoad
{
    self.tabBarItem = [[UITabBarItem alloc] initWithTitle:@"Playlists" image:[UIImage imageNamed:@"music.png"]
tag:0]
}
```
Creating Tab Bar Items

- (void)viewDidLoad
{
    self.tabBarItem = [[UITabBarItem alloc] initWithTabBarSystemItem:UITabBarSystemItemBookmarks tag:0]
}

More View Controllers

- What happens when a tab bar controller has too many view controllers to display at once?
  - “More” tab bar item displayed automatically
  - User can navigate to remaining view controllers
Custom Tab Bar Controller Demo

Combining Approaches
Tab Bar + Navigation Controllers

![Image of Tab Bar and Navigation Controllers]

**Navigation Controller** → **ViewController**

**Tab Bar Controller**

**Navigation Controller** → **ViewController**

**Navigation Controller** → **ViewController**

**Navigation Controller** → **ViewController**
Nesting Navigation Controllers

- Create a tab bar controller
  ```
  tabBarController = [[UITabBarController alloc] init];
  ```

- Create each navigation controller
  ```
  navController = [[UINavigationController alloc] init];
  [navController pushViewController:firstViewController animated:NO];
  ```

- Add them to the tab bar controller
  ```
  tabBarController.viewControllers = [NSArray arrayWithObjects:
  navController,
  anotherNavController,
  someViewController,
  nil];
  ```

Final Project Examples