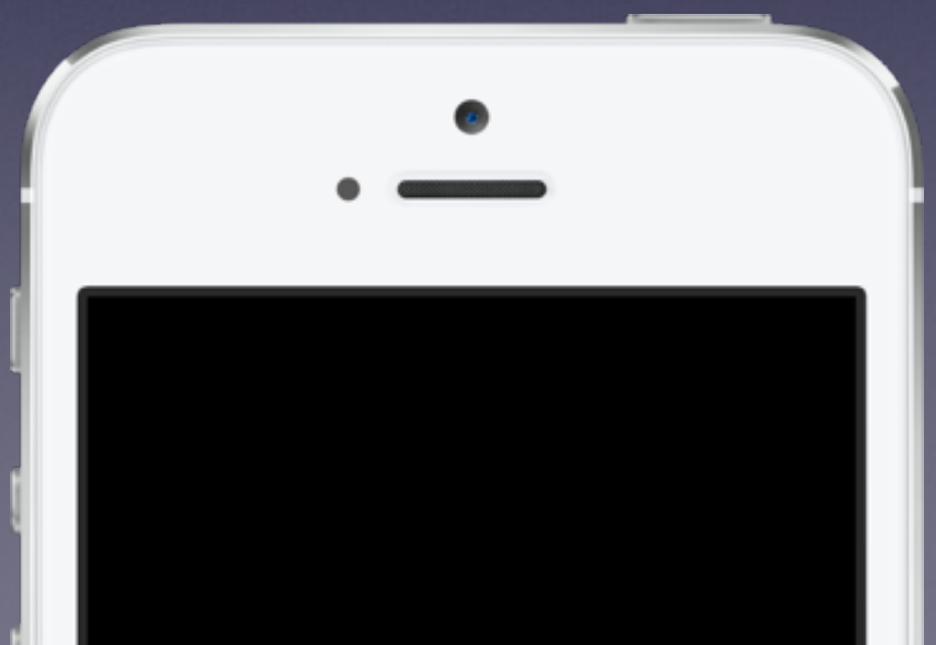
COMS W3101: Programming for iOS

Michael Vitrano



Views and View Controllers, part 1

- Understanding the View Controller Life-Cycle
- Working with view's programmatically
- UITableView
- Demo: Creating a custom UIViewController that displays a UITableView

Interface Builder

- Interface Builder the visual interface designer that is built into Xcode
- It provides two mechanisms for designing the interfaces associated with UIViewController subclasses
  - NIBs are a canvas for building the view for one VC
  - Storyboards act as a canvas for your entire sections of your application and define the relationships between multiple VCs

VC Life-Cycle

- The VC's view is initialized lazily. View creation happens in **-loadView**.
- In VC's created from a NIB or Storyboard, **-loadView** will unarchive the view described in the NIB
- If there is no NIB, the default implementation of **-loadView** creates an empty **UIView** to be the VC's view

VC Life-Cycle

- In VCs without a NIB or Storyboard, **-loadView** is where you should create your views programmatically
- Regardless of how the VC was created, **-viewDidLoad** is called after the VC's view has been loaded
  - **-viewDidLoad** is where additional configuration of your view controller should be done
- After **-viewDidLoad**, your VC should be completely configured and ready for use

VC Life-Cycle

- Before the VC's view is made visible, **-viewWillAppear:(BOOL)animated** is called
  - Useful for refreshing a view's data source before the view is presented to the user
- Once the VC's view is visible, **-viewDidAppear:(BOOL)animated** is called
  - Useful for displaying an alert to a user once the VC has been displayed to the user
- The default implementation of both of these methods do nothing

VC Life-Cycle

- Before the VC's view is removed from the hierarchy or is hidden, **-viewWillDisappear:(BOOL)animated** is called
- When the VC's view is finished being removed or hidden, **-viewDidDisappear:(BOOL)animated** is called
  - Useful for canceling network requests that are related to a specific VC
- The default implementation of both of these methods do nothing

UIView

- UIView is the base class for all views in iOS
- It handles both the rendering of content and interaction with that content
- There are a number of subclasses provided by UIKit, including **UIButton** and **UIImageView** that provide implementations of core functionality
- UIView provides hooks for drawing custom content to the screen and animating it
- Views can add instances of **UIGestureRecognizer** to handle complex multitouch interactions

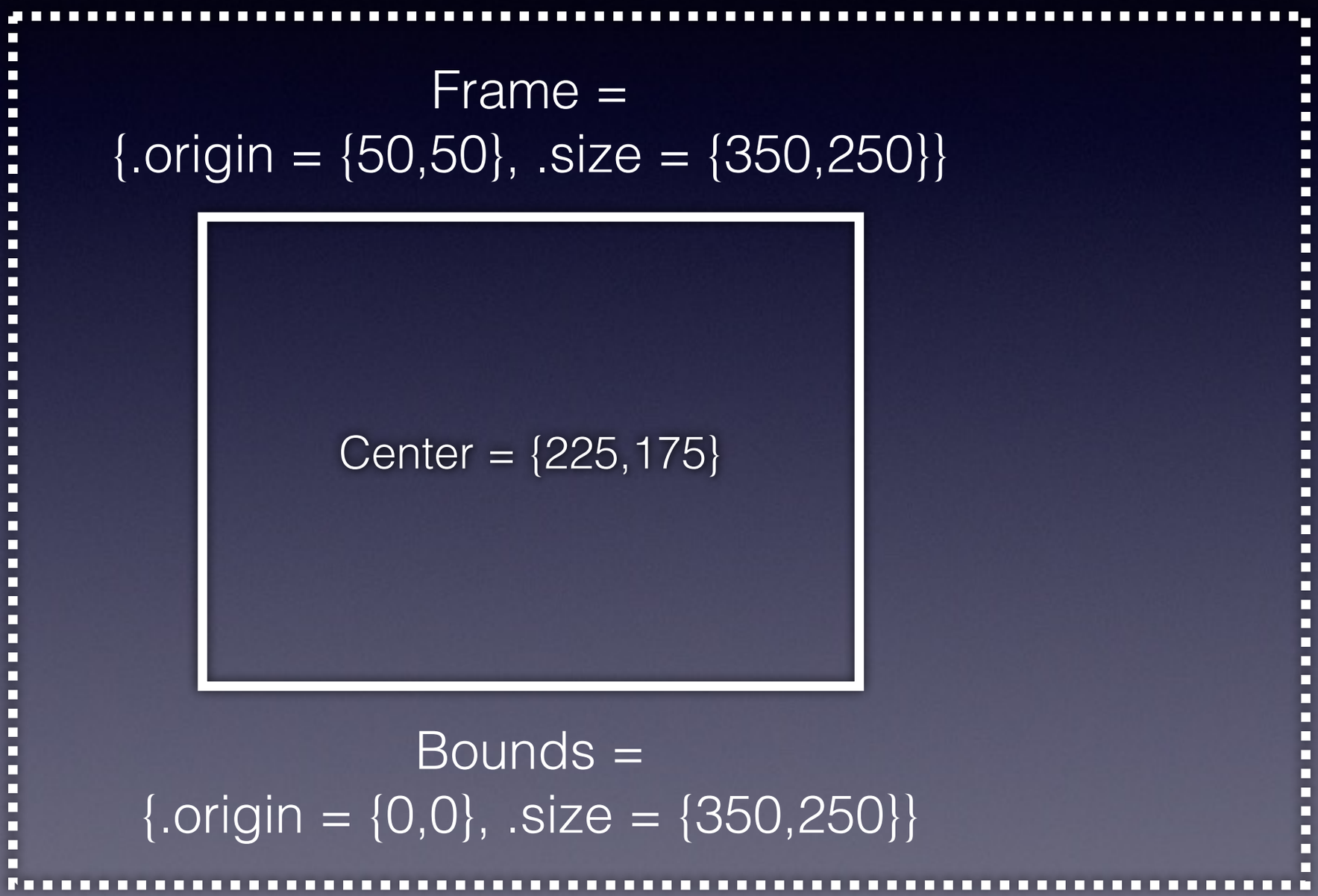
Defining View Geometry

- A set of C-style structs are central to defining the geometry:
  - **CGPoint -** `{CGFloat x; CGFloat y;}`
  - **CGSize -** `{CGFloat width; CGFloat height;}`
  - **CGRect -** `{CGPoint origin; CGSize size;}`

View Geometry

- A view's geometry is defined by its **frame**, **bounds**, and **center** properties
  - The **frame** defines the origin and size of the view within the coordinate system of its **superview**
  - A view's **center** represents the center of the view within its **superview**'s coordinate system
  - The **bounds** defines the views position and size in its own coordinate system
  - The size of the **bounds** and **frame** are tied together

View Geometry



View Appearance

- **backgroundColor** defines the background color of a view
  - Defaults to nil, which renders as transparent
- **alpha** defines the transparency of your view
  - 1.0 is fully opaque, 0.0 is completely transparent
  - This includes all subviews
- **tintColor** defines the view's 'highlight' color
  - For example, this might determine the color of a button's icon

View Hierarchy

- You can add a view as a subview of another view by calling **-addSubview:** on the superview with the subview as the parameter
- Call **-removeFromSuperview** on a view to remove it from the view hierarchy
- A view's **superview** property will return the view it is embedded in or **nil** if its not in the hierarchy
- The **subviews** property returns an NSArray containing the subview contained within the view

View Layout

- In a custom UIView subclass, you have the ability to layout your custom subviews in **-layoutSubviews**
- Always layout subviews with relation to a view's **bounds** which represent the internal coordinate system
- Calling **setNeedsLayout** on a view marks it for layout by the OS

AutoLayout

- System for defining the layout of UI elements as a mathematical system of relationships called constraints
  - There are many types of constraints, including ones for view size, relative positioning, margin in relation to other views
- Far more flexible than autoresizing and requires less code than implementing **-layoutSubviews**
- Constraints can be defined both in code as well as using Interface Builder

UITableView

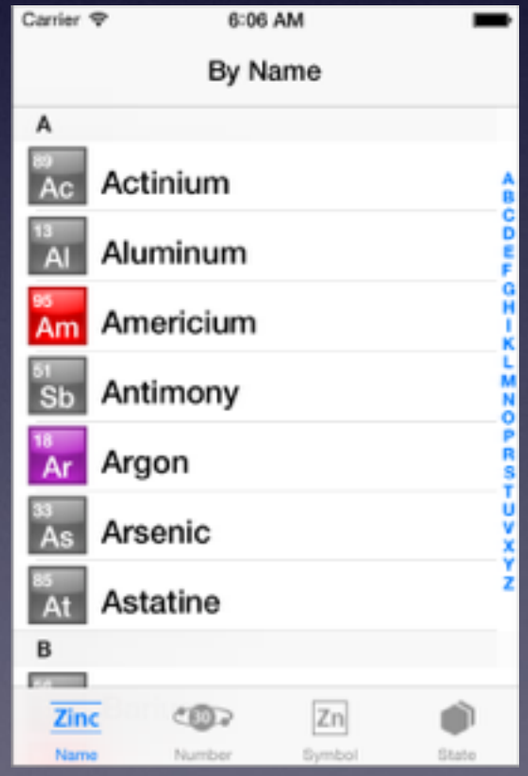
- A view that specializes in displaying information in a vertically scrolling list
- Individual cells in a tableview are instances of **UITableViewCell**
- TableViews have a **dataSource** that conforms to the **UITableViewDataSource** protocol
  - The **dataSource** is responsible for telling the TableView how many sections and how many rows are in each section of the table
  - The **dataSource** is also responsible for providing configured instances of a **UITableViewCell** for each item in the table

UITableView

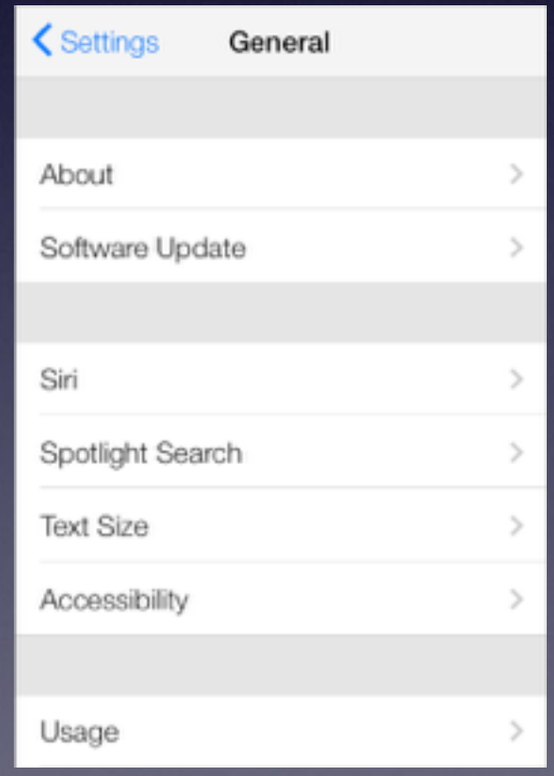
- TableViews also have a **delegate** property which is responsible for:
  - Responding to the selection of cells when a user interacts with the tableView
  - Configuring row reordering and editing
  - Configuring cell highlighting behavior

UITableView

- TableViews can come in two styles:



UITableViewStylePlain



UITableViewStyleGrouped

UITableViewDataSource

- Required Methods:

```
- (NSInteger)tableView:(UITableView *)tableView
 numberOfRowsInSection:(NSInteger)section

// This method returns the view to display for the section/row specified by
indexPath
- (UITableViewCell *)tableView:(UITableView *)tableView
        cellForRowAtIndexPath:(NSIndexPath *)indexPath
```

- Optional Methods:

```
// If this method is not implemented, 1 section is the default
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView

// This method returns the string to display as the header for the specified section
- (NSString *)tableView:(UITableView *)tableView
titleForHeaderInSection:(NSInteger)section

 // This method returns the string to display as the footer for the specified
section
- (NSString *)tableView:(UITableView *)tableView
titleForFooterInSection:(NSInteger)section
```

UITableViewDelegate

- All methods are optional in this protocol:

```
// Provides a hook to do custom actions when a row is selected
- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath;

// Defaults to the -rowHeight property of UITableView if not implemented
- (CGFloat)tableView:(UITableView *)tableView
heightForRowAtIndexPath:(NSIndexPath *)indexPath;

// These methods allow for completely custom views to be show in the
// header and footer areas of sections

- (UIView *)tableView:(UITableView *)tableView
viewForHeaderInSection:(NSInteger)section;

- (UIView *)tableView:(UITableView *)tableView
viewForFooterInSection:(NSInteger)section;
```