iPhone Application Programming
Lecture 4: User Interface Design

View Concepts

- SDK provide many types of Views to show your content
- At run-time Views are organized as a tree
- Use Interface Builder to design your UI and connect it to code
- Geometry of Views are determined by constraints
Finding the Right View

A label displays static text.

- Displays any amount of static text
- Doesn’t allow user interaction except, potentially, to copy the text

Use a label to name or describe parts of your UI or to provide short messages to the user. A label is best suited for displaying a relatively small amount of text.

Take care to make your labels visible. It’s best to support Dynamic Type and use the `UIFont` method `preferFontForTextStyle` to get the font for display in a label. If you choose to use custom fonts, don’t sacrifice clarity for fancy lettering or showy colors. (For guidelines and more information on Dynamic Type, see “Text Styles” in “User Interface Programming—iOS.”)

View Concepts

✓ SDK provides many types of views to show your content
- At runtime views are organized as a tree
- Use Interface Builder to design your UI and connect it to code
- Geometry of views are determined by constraints

Demo: Hacking Calendar

```swift
(lldb) po [[UIWindow keyWindow] recursiveDescription]
<UIWindow
   | <UILayoutContainerView
   |    | <UINavigationTransitionView
   |    |    | <UIViewControllerWrapperView
   |    |    |    | <EKCalendarItemEditorTableView: 0xf870200; baseClass = UITableView;
   |    |    |    |    | <UITableViewWrapperView
   |    |    |    |    |    | <UITableViewCell: 0xb19ad90
   |    |    |    |    |    |    ... 
   |    |    | <UINavigationBar
   |    |    |    | <UILabel
   |    |    | <UINavigationButton
   |    |    |    ... 
   ... 
   <UINavigationBar
   ... 
   <UINavigationItemView
   ... 
   <UILabel
   ... 
}
(lldb) expr ((UIView *)0xb19ad90).backgroundColor = [UIColor redColor]
(lldb) expr ((UIView *)0xf870200).backgroundColor = [UIColor greenColor]
```
View Concepts

✓ SDK provide many types of Views to show your content
✓ At run-time Views are organized as a tree
• Use Interface Builder to design your UI and connect it to code
• Geometry of Views are determined by constraints

Interface Builder

• Graphical tool to layout user interfaces
• Create the widget hierarchy
• Set attributes of widgets
• Set up connections between the widgets
• Store these informations in nib files

The Anatomy of a xib File
The Source of a xib

```xml
<?xml version="1.0" encoding="UTF-8"?>
<archive type="com.apple.InterfaceBuilder3.CocoaTouch.XIB" version="8.00">
  <data>
    <int key="IBDocument.SystemTarget">1280</int>
    <string key="IBDocument.SystemVersion">11C74</string>
    <string key="IBDocument.InterfaceBuilderVersion">1938</string>
    <string key="IBDocument.AppKitVersion">1138.23</string>
    <string key="IBDocument.HIToolboxVersion">567.00</string>
    <object class="NSMutableDictionary" key="IBDocument.PluginVersions">
      <string key="NS.key.0">com.apple.InterfaceBuilder.IBCocoaTouchPlugin</string>
      <integer value="933" key="NS.object.0"/>
    </object>
    <array key="IBDocument.IntegratedClassDependencies">
      <string>IBUITextView</string>
      <string>IBUISwitch</string>
      <string>IBUIButton</string>
      <string>IBUIView</string>
      <string>IBUILabel</string>
      <string>IBProxyObject</string>
    </array>
    <array key="IBDocument.PluginDependencies">
      <string>com.apple.InterfaceBuilder.IBCocoaTouchPlugin</string>
    </array>
    <object class="NSMutableDictionary" key="IBDocument.Metadata">
      <string key="NS.key.0">PluginDependencyRecalculationVersion</string>
      <integer value="1" key="NS.object.0"/>
    </object>
    <array class="NSMutableArray" key="IBDocument.RootObjects">
      <object class="IBProxyObject" id="841351856">
        <string key="IBProxiedObjectIdentifier">IBFilesOwner</string>
        <string key="targetRuntimeIdentifier">IBCocoaTouchFramework</string>
      </object>
      <object class="IBProxyObject" id="371349661">
        <string key="IBProxiedObjectIdentifier">IBFirstResponder</string>
        <string key="targetRuntimeIdentifier">IBCocoaTouchFramework</string>
      </object>
      <object class="IBUIView" id="474857037">
        <reference key="NSNextResponder"/>
        <int key="NSvFlags">292</int>
        <array class="NSMutableArray" key="NSSubviews">
          <object class="IBUITextView" id="694905917">
            <reference key="NSNextResponder" ref="474857037"/>
            <int key="NSvFlags">274</int>
            <string key="NSFrame">{{20, 20}, {280, 385}}</string>
            <reference key="NSSuperview" ref="474857037"/>
          </object>
        </array>
      </object>
    </array>
  </data>
</archive>
```

Laying out the User Interface

- The library contains all UI Widgets
- Drag them to your view
- See instantly what your UI looks like
- Test your UI in the iPhone Simulator

Set Widget Attributes

Connecting Widgets and Code

- IBActions
  - Tags a method as a target for an interface action
- IBOutlets
- IBOutletCollection
  - Variables to populate with objects from a nib file
Connecting Widgets and Code

@interface iCalcView: UIView {
}
-(IBAction)addDigit:(id)sender;
-(IBAction)calculateResult:(id)sender;

//declared properties
//textfield
@property (nonatomic, weak) IBOutlet UITextField *textfield;

//operators: + =
@property (nonatomic, weak) IBOutlet UIButton *operatorPlus;
@property (nonatomic, weak) IBOutlet UIButton *operatorEqual;

//digits
@property (nonatomic, weak) IBOutletCollection (UIButton) NSArray *digitButtons;

Interface Builder Demo

View Concepts

✓ SDK provide many types of Views to show your content
✓ At run-time Views are organized as a tree
✓ Use Interface Builder to design your UI and connect it to code
  • Geometry of Views are determined by constraints

Auto Layout

• Preferred layout management
• Allows you to create views that work both in portrait and landscape mode
• Available in iOS 6 and higher
• Spatial relationships expressed by constraints
Auto Layout Constraints

- Constraints are mathematical expressions
  - $\leq$, $\geq$
- Constraints have a priority level
- The runtime tries to solve the system of equations

Adding Constraints

View Concepts

- SDK provides many types of Views to show your content
- At run-time Views are organized as a tree
- Geometry of Views are determined by constraints
- Use Interface Builder to design your UI and connect it to code

UIView

- Defines a rectangular area on the screen
- Two responsibilities
  - Render content
  - React to user input
  - Manage subviews
- Layout as view hierarchy
Clarke to shut six prisons to save cash

Move as 'legal constraints' on job-watching

Richard Foott, Rachel Schneider

The government is facing growing pressure to save money. Prison closures were previously announced, but the new closures, which are expected to save over £1 billion, will be the largest in modern times. The closures will affect six prisons, including the notorious Maze prison, which has been the site of numerous disturbances in recent years.

Right From Fire Mountain. Villagers flee from deadly volcanic ash cloud

The Times

 hüllt

Die Welt HD

View Hierarchy

- NSObject
- UIResponder
- UIView
- UIWindow
- UILabel
- UIPickerView
- UIProgressView
- UIActivityIndicatorView
- UIImageView
- UITabBar
- UIToolbar
- UINavigationBar
- UITableViewCell
- UIActionSheet
- UIAlertView
- UISearchBar
- UIWebView
- UIScrollView
- UITableView
- UITextView
- UIControl
- UIButton
- UIDatePicker
- UIPageControl
- UISegmentedControl
- UITextField
- UISlider
- UISwitch

Die Welt HD

- The Times
- The Times
- The Times
- The Times
- The Times
- The Times
UI Widgets

- Containers: UIScrollView
- Controls: UIButton
- Display views: UIImageView
- Text and web views: UITextView
- Alert views: UIAlertView
- Navigation views: UITabBar
- UIWindow

Control Widgets

- Manages typically one screen
- Flushes the view on low-memory situations
- Resizes the view on orientation change
- Creates modal views on top of the current view

View Categories: Examples

- Containers: UIScrollView
- Controls: UIButton
- Display views: UIImageView
- Text and web views: UITextView
- Alert views: UIAlertView
- Navigation views: UITabBar
- UIWindow

UIKitViewController