

Mobile Application Development

MAS 490: Theory and Practice of Mobile Applications

Professor John F. Clark

What is Interface Builder?

- Interface Builder is a software development application for Apple's Mac OS X operating system.
- It is part of Xcode, the Apple developer's toolkit
- Interface Builder allows Cocoa and Carbon developers to create interfaces for applications using a graphical user interface (GUI).
- The resulting interface is stored as a .nib file, short for NeXT Interface Builder
- As of Xcode 4, Interface Builder is no longer a stand-alone application and is fully integrated into Xcode

History of Interface Builder

- Interface Builder dates back to 1986. It was originally written in LISP, which is the second-oldest high-level programming language still in use. It was deeply integrated into the Macintosh toolbox in the days of Mac OS 8.
- It was introduced to Steve Jobs when he was running NeXT. By 1988 it was incorporated in NeXTSTEP.
- It was the first commercial application that allowed buttons, menus, and windows to be placed in an interface using a mouse.

Tim Berners-Lee



Tim uses a NeXT computer to design the first web browser



NeXTSTEP Desktop



How does it work?

- Interface Builder provides collections (or palettes) of user interface objects (text fields, data tables, sliders, and pop-up menus, for example) to the Objective-C programmer.
- The palettes are extensible, meaning you can customize and develop new objects that can be added to new or existing palettes.
- To build an interface, the developer just drags and drops objects into a window or menu. Actions that the objects can perform are connected to targets in the code and outlets (pointers) declared in the code are connected back to objects.

How does it work? Part II

- Interface Builder saves an application's interface as a bundle that contains the interface objects and relationships used in the application.
- These objects are archived into either an XML file or a NeXT-style property list file with a .nib extension.
- Upon running an application, the proper NIB objects are unarchived, connected into the binary of their owning application, and awakened.
- NIBs are often referred to as *freeze dried* because they contain the archived objects themselves, ready to run.

Basic Tools

- Xcode developer environment
 - For writing code
- Interface Builder
 - GUI for designing interfaces
- Connections:
 - Xcode has IBOutlet and IBAction types to connect Interface Builder elements to code objects.

Some User Interface (UI) Elements

- UINavigationController (optional)
 - ViewController
 - View
 - Image
 - Label
 - Button
 - TextField

Varieties of “C” Code

- Objective-C basics:
 - Simple method, no parameters:
 - [robot stand];
 - robot.stand();
 - Method with one parameter:
 - [robot walkDistance:(int)distance];
 - robot.walk(int distance);
 - Method with two parameters:
 - [robot walkDistance:(int)distance inDirection:(float)direction];
 - robot.walk(int distance, int direction);

View Controller

- Most interface logic belongs in a view controller subclass
- Event Driven
 - Init/initWithNibName:Bundle:
 - ViewDidLoad
 - ViewWillAppear:
 - ViewDidAppear:
 - ViewWillDisappear:
 - ViewDidDisappear:

Common Usage

- ViewDidLoad
 - Called once when view is finished initializing and is added to the view stack
 - Use this for code that you want to run only once before the user sees anything
 - Example:
 - Set the title of the view
 - `[self setTitle:@"My View"];`

Common Usage, Part II

- ViewWillAppear
 - Called just before the view becomes visible, can be called multiple times (back button)
 - Use this for code that you want to run every time the view is displayed
 - Example:
 - Reload a page's dynamic contents
 - `[self setViewCounter:viewCounter + 1];`
 - `self.counterLabel.text = self.viewCounter;`

Common Usage, Part III

- ViewWillDisappear
 - Called after the view becomes invisible, can be called multiple times
 - Use this for code that you want to run every time the view is gone
 - Example
 - Stop refreshing a timer
 - `[self stopMyTimer];`

A More Complicated Example

- Let's suppose a View Controller contains six labels.
 - The content of three of the labels is dynamic and could be longer than one line.
 - The code must resize the labels and move down the remaining labels.


```
// Category for UIView
@interface UIView (UIKitExtensions)
- (void)moveDown:(int)pixels;
- (void)updateSize:(CGSize)newSize;
@end

@implementation UIView (UIKitExtensions)

- (void)moveDown:(int)pixels {
    self.frame = CGRectMake
(self.frame.origin.x, self.frame.origin.y +
pixels, self.frame.size.width, self.frame.size.height);
}

- (void)updateSize:(CGSize)newSize {
    self.frame = CGRectMake
(self.frame.origin.x, self.frame.origin.y, newSize.width,
newSize.height);
}

@end
```

```
int labelWidth = challenges.frame.size.width;
int labelHeight = challenges.frame.size.height;
int extraHeight = 0;
int labelMaxHeight = 2000;
int originalHeight;
UIFont *font = challenges.font;
CGSize size;

challenges.text = issue.challenges;
contributingFactors.text = issue.contributingFactors;
consequences.text = issue.consequences;

// size challenges
originalHeight = challenges.frame.size.height;
size = [challenges.text sizeWithFont:font
        constrainedToSize:CGSizeMake(labelWidth, labelMaxHeight)];
size.height = (size.height < labelHeight ? labelHeight : size.height);
[challenges updateSize:size];

// update next label
extraHeight += size.height - originalHeight;
[contributingFactorsHeader moveDown:extraHeight];
[contributingFactors moveDown:extraHeight];

//...continued
```

```
// size contributingFactors
originalHeight = contributingFactors.frame.size.height;
size = [contributingFactors.text sizeWithFont:font
        constrainedToSize:CGSizeMake(labelWidth, labelMaxHeight)];
size.height = (size.height < labelHeight ? labelHeight : size.height);
[contributingFactors updateSize:size];

// update next label
extraHeight += size.height - originalHeight;
[consequencesHeader moveDown:extraHeight];
[consequences moveDown:extraHeight];

// size consequences
originalHeight = consequences.frame.size.height;
size = [consequences.text sizeWithFont:font
        constrainedToSize:CGSizeMake(labelWidth, labelMaxHeight)];
size.height = (size.height < labelHeight ? labelHeight : size.height);
[consequences updateSize:size];
```