CS 696 Mobile Application Development
Fall Semester, 2010
Doc 8 Segment Control, Alerts & Action Sheets
Sep 23, 2010

References

iPhone Human Interface Guideline, http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html

Various UI class documentation

Beginning iPhone 3 Development: Exploring the iPhone SDK by Jeff LaMarche, and David Mark, Chapter 4

iPad

1024 by 768 pixels

iOS 3.2

iPhone views & controls work

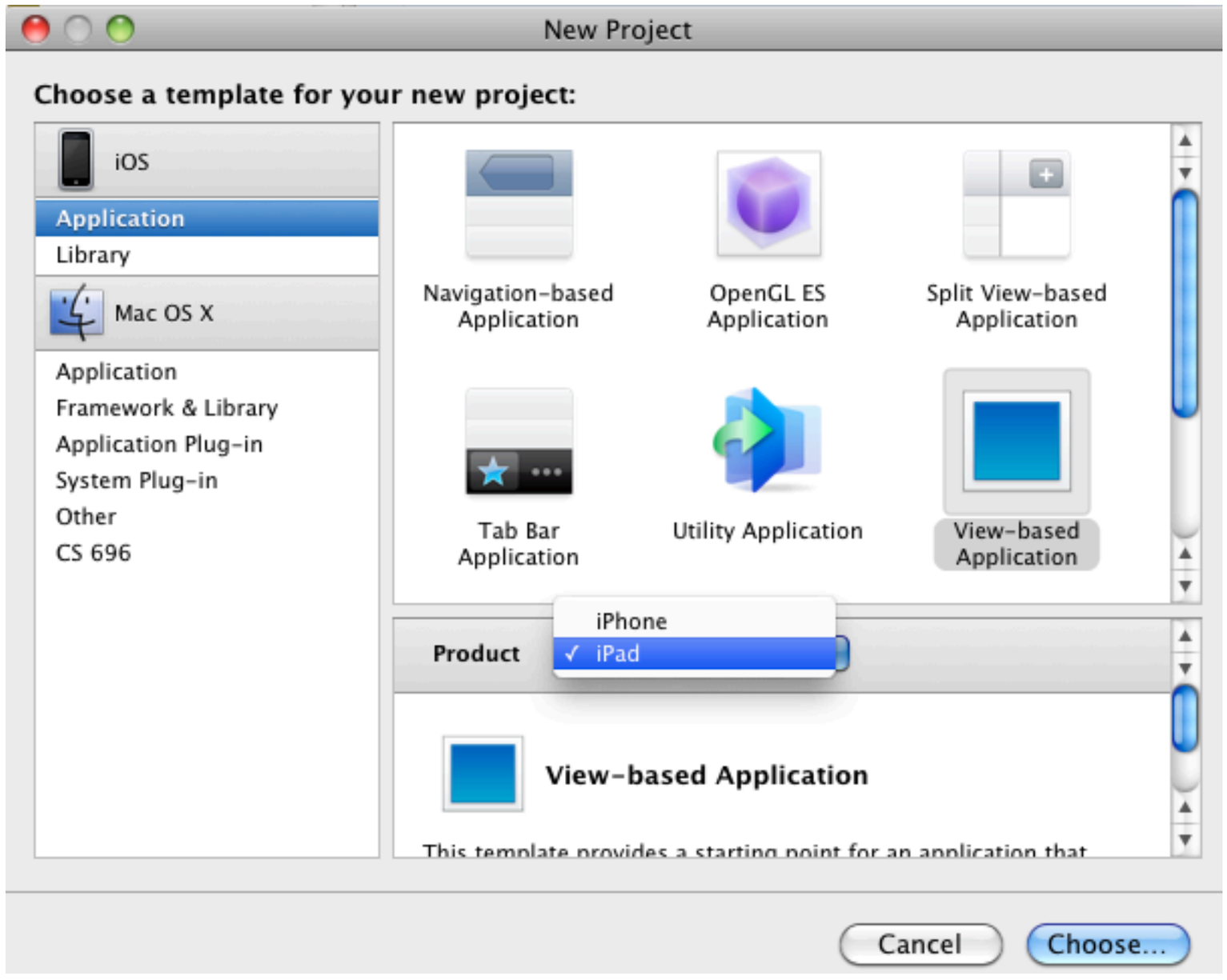
Added

- Split views
- Popovers

Toolbars top or bottom

Can replace keyboard with custom input

Creating Project

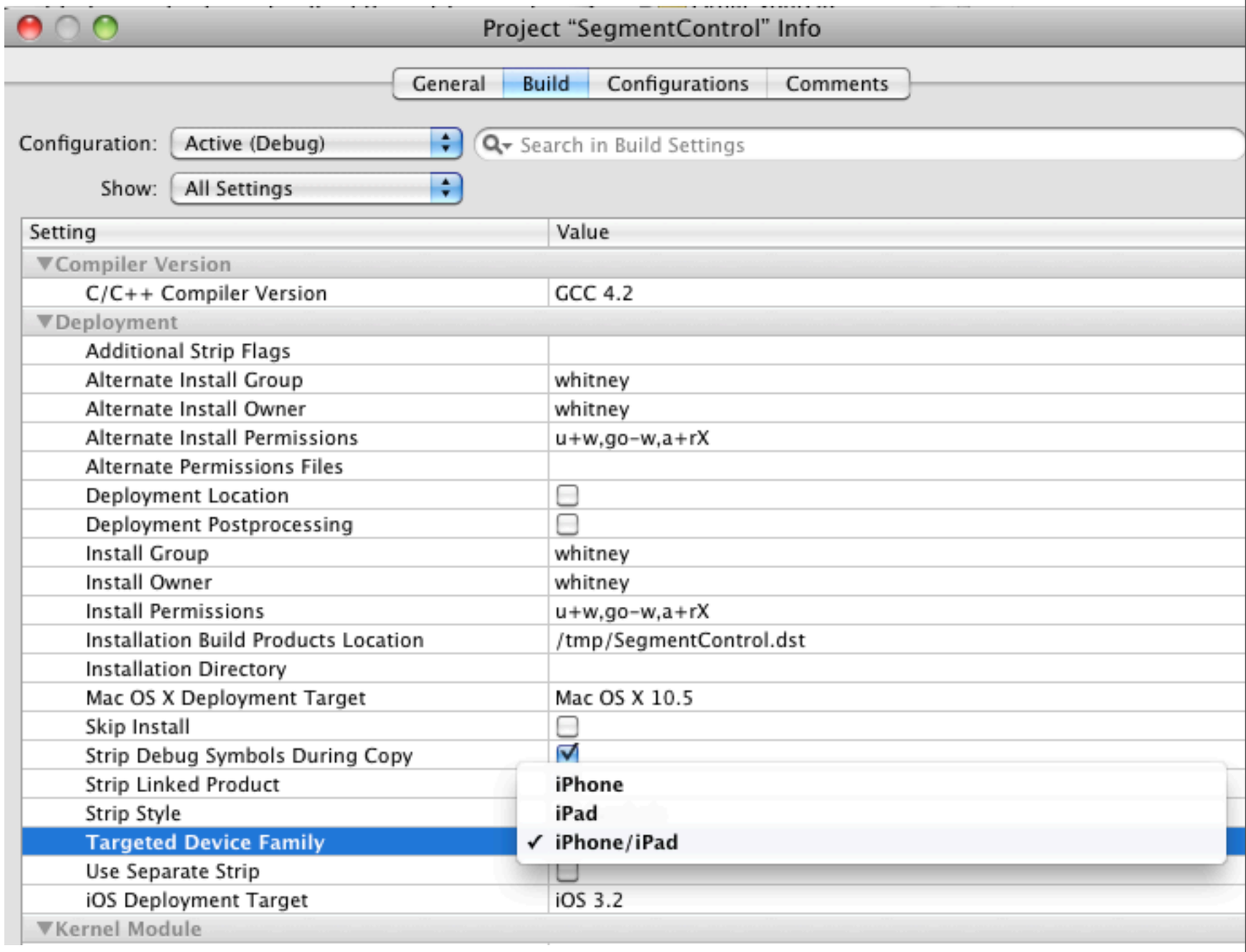


Targeted Device

iPhone/ipad
  universal binary

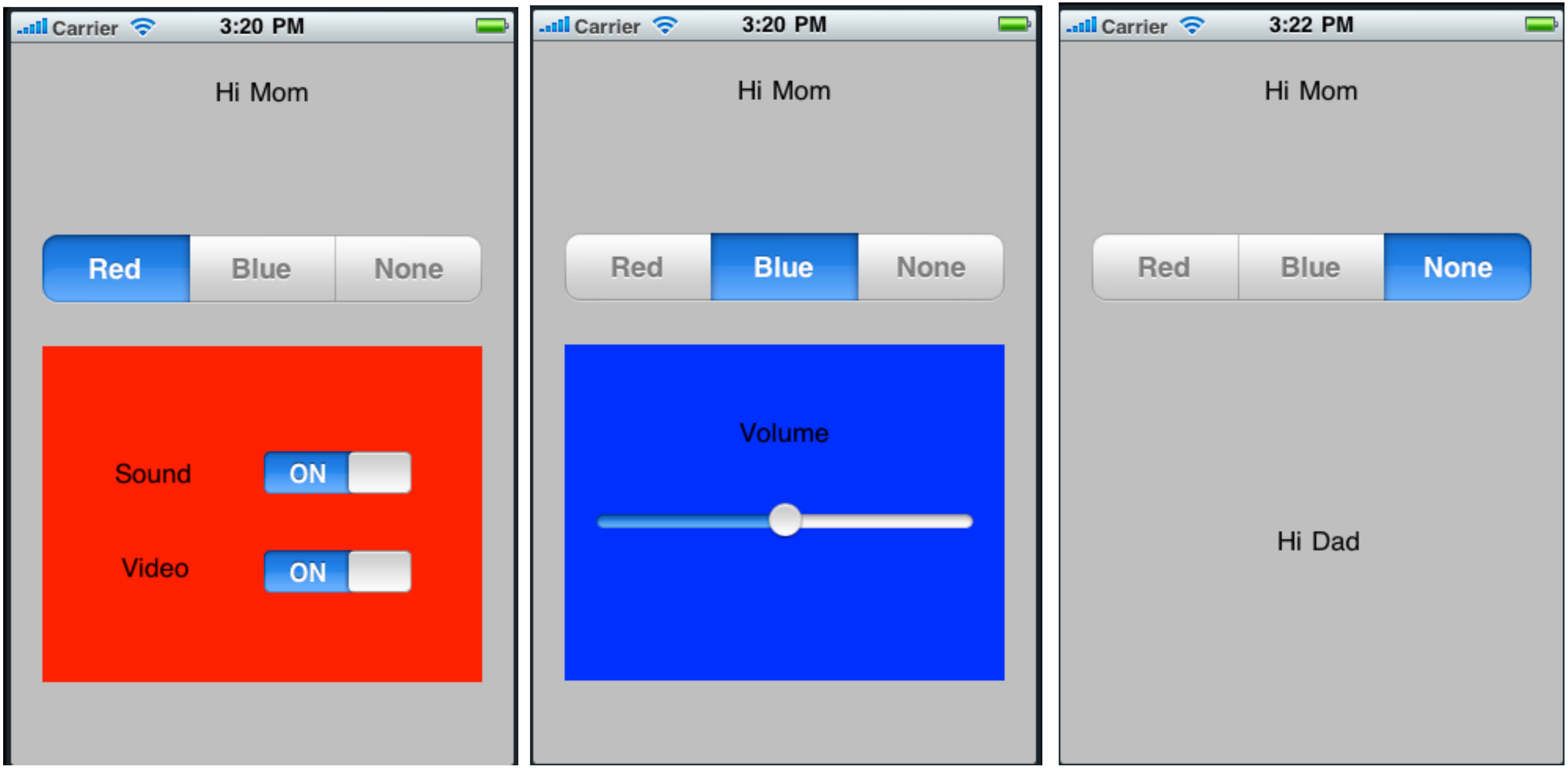
iPhone
  runs on ipad
  Uses iPhone screen

iPad
  does not run on iPhone



Segment Control

Segments in control controls together

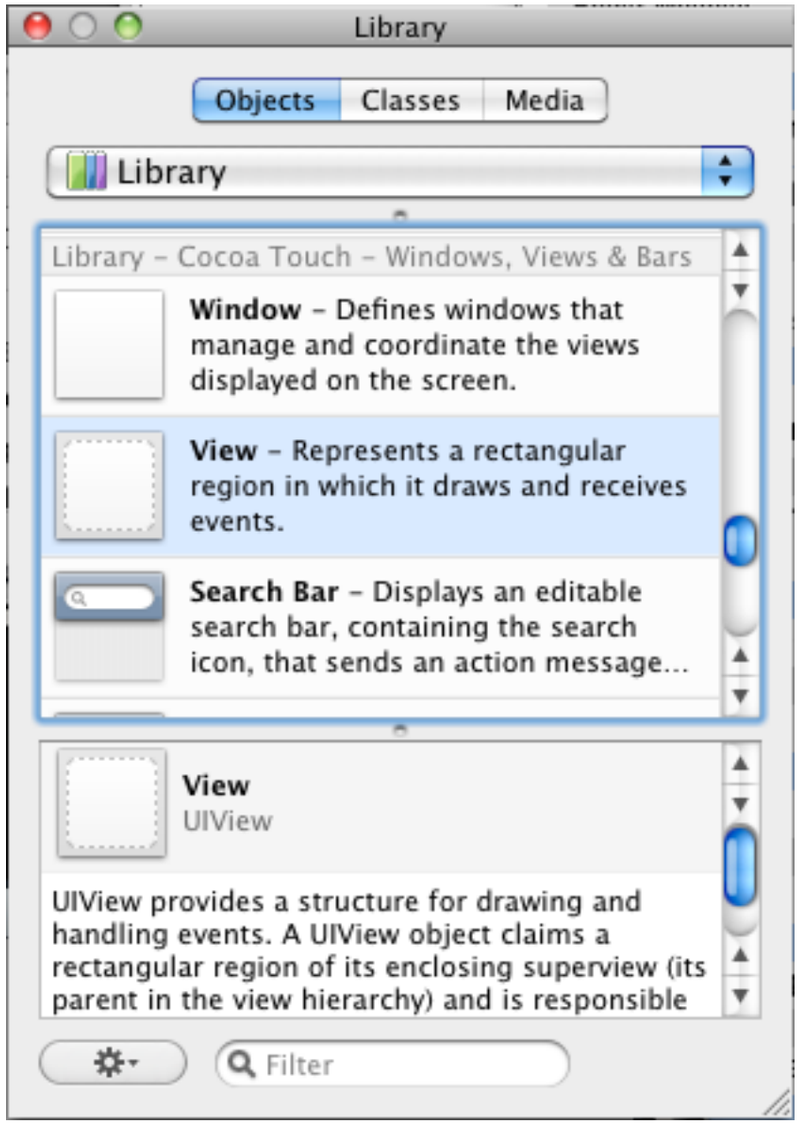


UIView

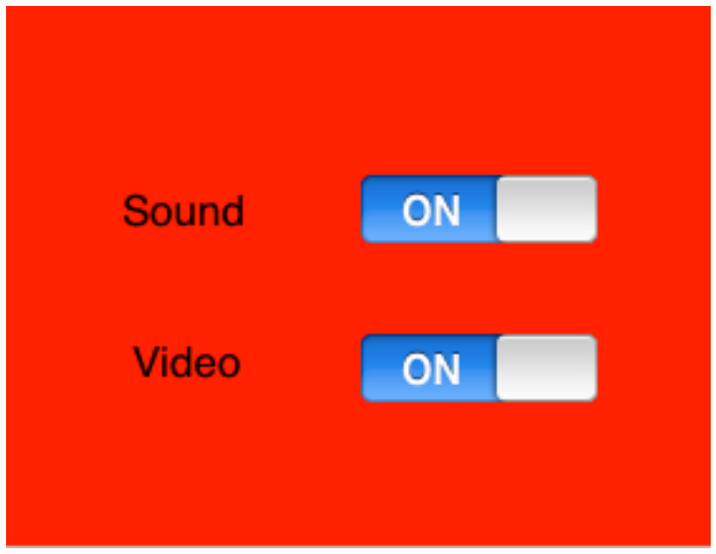
Container for views

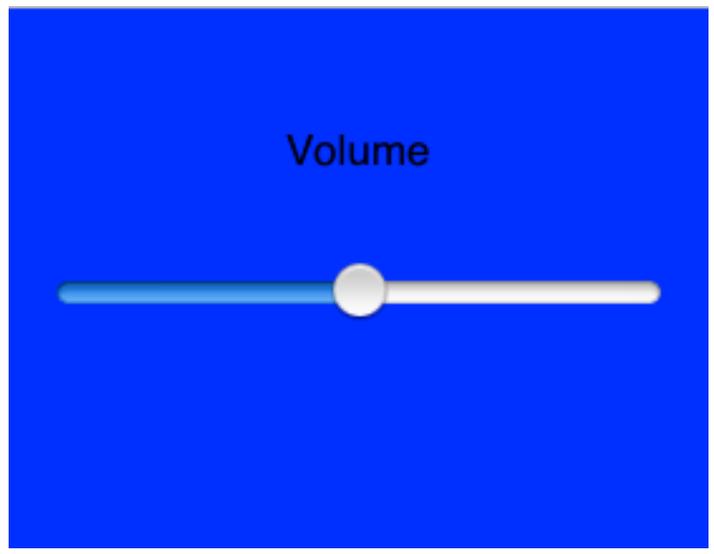
Subclass it for custom content

When view is hidden all views it contains are hidden



Nested Views

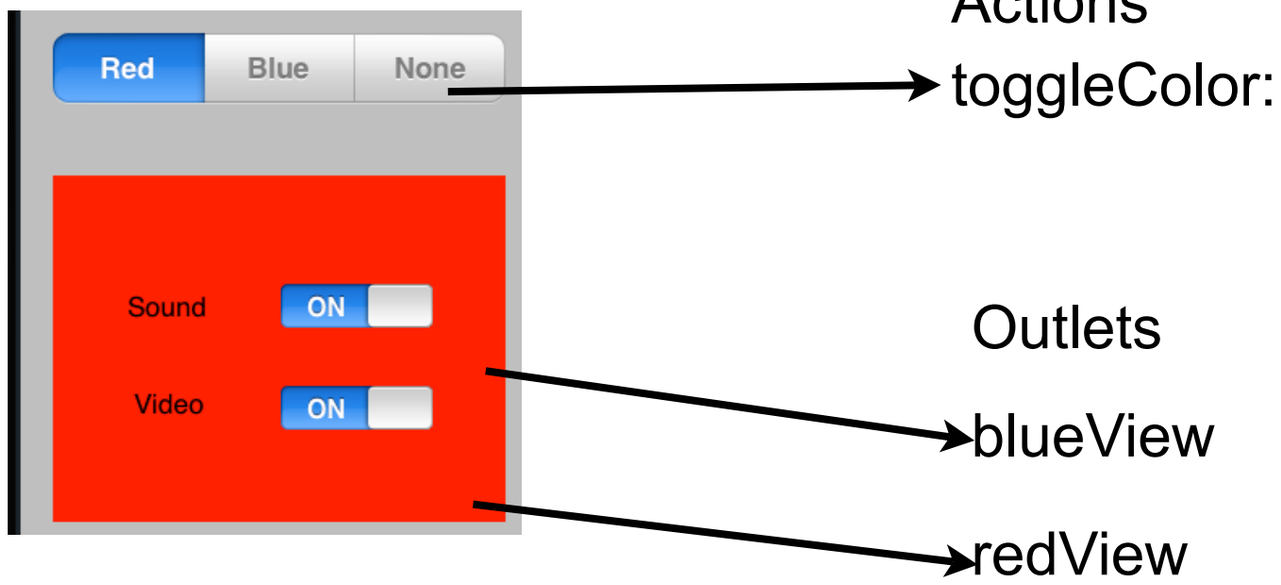




UIView with red background
Contains
- 2 labels
- 2 switches
- Blue UIView (top level)

UIView with blue background
Contains
- Label
- Slider

Connections




```
@interface SegmentControlViewController : UIViewController { }
@property (nonatomic, retain) IBOutlet UIView * blueView;
@property (nonatomic, retain) IBOutlet UIView * redView;
- (IBAction) toggleColor: (id) sender;
@end
```

SegmentControlViewController

```
#import "SegmentControlViewController.h"
#define RED_INDEX  0
#define BLUE_INDEX  1

@implementation SegmentControlViewController

@synthesize  blueView;
@synthesize  redView;
```

SegmentControlViewController

```
- (IBAction) toggleColor: (id) sender {
    switch ([sender selectedSegmentIndex]) {
        case RED_INDEX:
            [self showRed];
            break;
        case BLUE_INDEX:
            [self showBlue];
            break;
        default:
            [self hideBlueRed];
            break;
    }
}
```

SegmentControlViewController

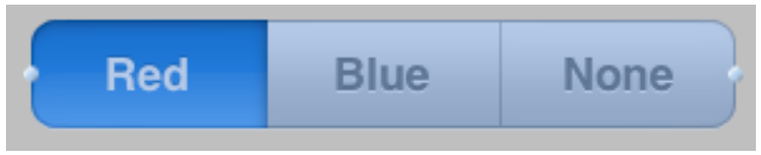
```
- (void) showRed {
	blueView.hidden = YES;
	redView.hidden = NO;
}

- (void) showBlue {
	//blue is the top view inside of red
	blueView.hidden = NO;
	redView.hidden = NO;
}

- (void) hideBlueRed {
	//blue is the top view inside of red
	//blue is hidden when red is
	redView.hidden = YES;
}
```

SegmentControlViewController

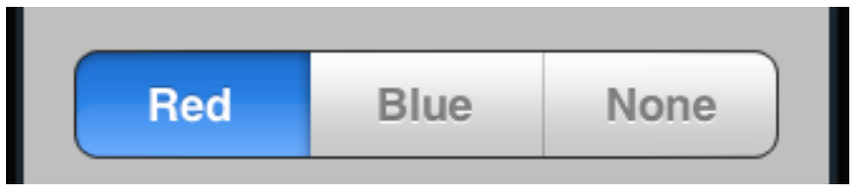
```
- (void) dealloc {
        [blueView release];
        [redView release];
}
```

Properties
segmentedControlStyle
tintColor

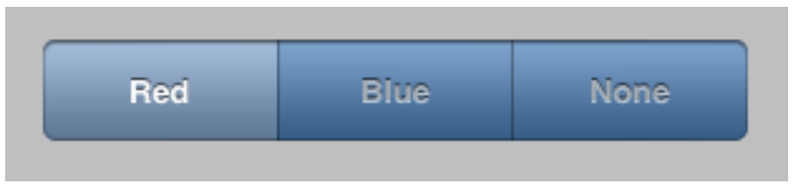
plain **Options**



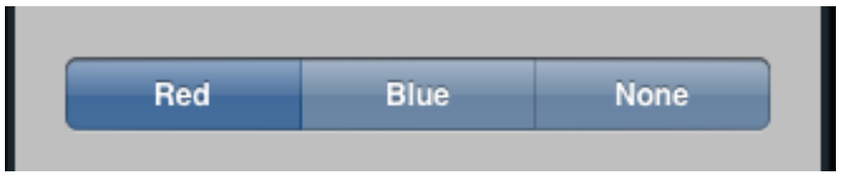
bordered



bezeled

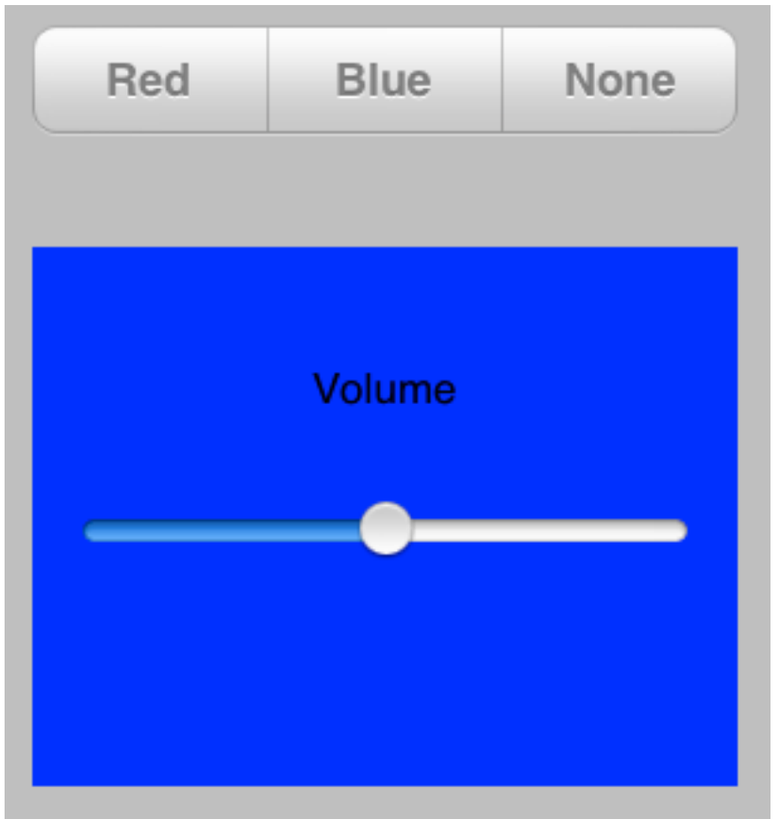


bar



bar with red tint

Momentary

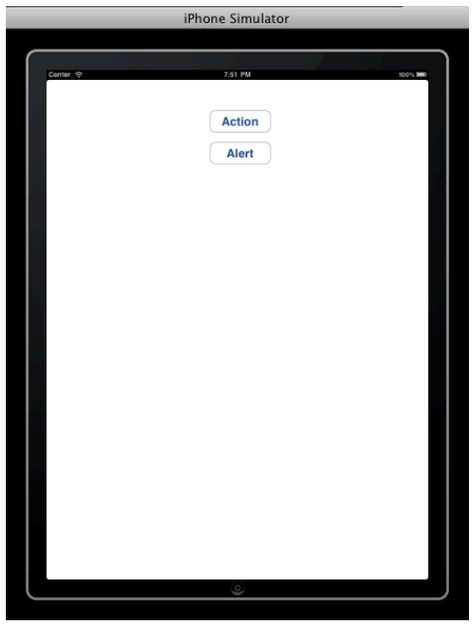


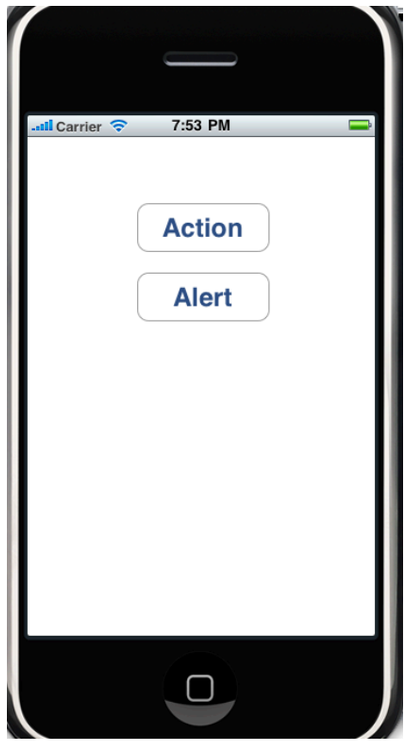
Linking actions in code

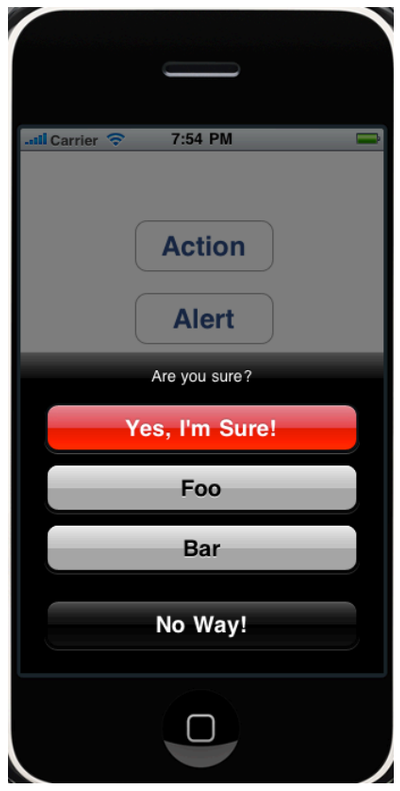
```
@property (nonatomic, retain) IBOutlet UISegmentedControl * colorSelector;
```

```
- (void)viewDidLoad {
    [super viewDidLoad];
        [colorSelector
            addTarget:self
            action:@selector(toggleColor:)
            forControlEvents:UIControlEventValueChanged];
}
```

Alert & Action Sheets







Alerts & Actionsheets

modal - user has to dismiss them

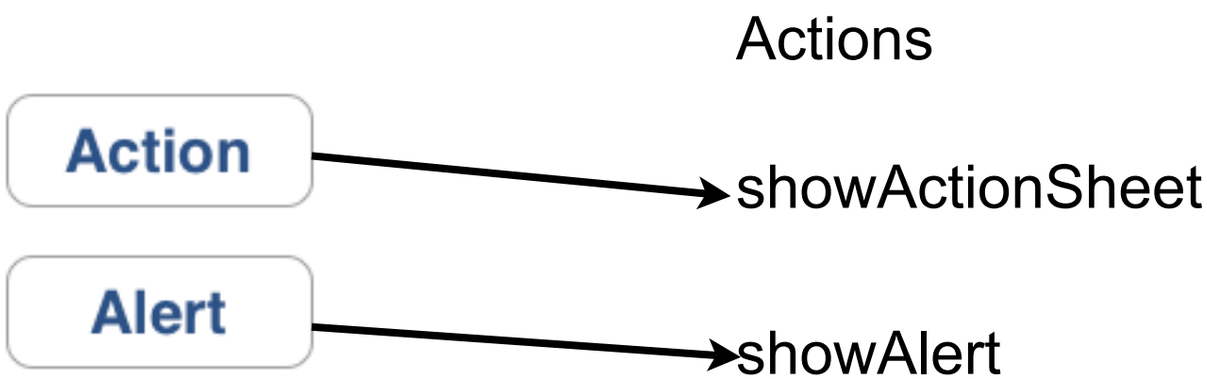
Alert

Gives information about problem or change

Action sheet

Gives additional choices related to current action

The Connections




```
@interface AlertActionSheetViewController : UIViewController
<UIActionSheetDelegate, UIAlertViewDelegate>{ }
- (IBAction) showAlert;
- (IBAction) showActionSheet;
@end
```

Creating the ActionSheet

```
- (IBAction) showActionSheet {
    UIActionSheet *actionSheet = [[UIActionSheet alloc]
                                  initWithTitle:@"Are you sure?"
                                  delegate:self
                                  cancelButtonTitle:@"No Way!"
                                  destructiveButtonTitle:@"Yes, I'm Sure!"
                                  otherButtonTitles:@"Foo",@"Bar",nil];
    actionSheet.actionSheetStyle = UIActionSheetStyleBlackOpaque;
    [actionSheet showInView:self.view];
    [actionSheet release];
}
```

Responding to users selection

```
- (void)actionSheet:(UIActionSheet *)actionSheet didDismissWithButtonIndex:(NSInteger)
buttonIndex {
	NSLog(@"action");
	if (buttonIndex == [actionSheet cancelButtonIndex]){
		NSLog(@"cancelled");
		return;
	}
	NSLog(@"User selected %@", [actionSheet buttonTitleAtIndex:buttonIndex]);
}
```

Showing the alert

```
- (IBAction) showAlert {
    UIAlertView *alert = [[UIAlertView alloc]
                          initWithTitle:@"This is an alert"
                          message: @"Hi mom"
                          delegate:self
                          cancelButtonTitle:@"Cancel"
                          otherButtonTitles:@"A",@"B",nil];
    [alert show];
    [alert release];
}
```

Responding to the Alert

```
- (void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:(NSInteger)buttonIndex {
    if (buttonIndex == [alertView cancelButtonIndex]){
        NSLog(@"cancelled");
        return;
    }
    NSLog(@"User selected %@", [alertView buttonTitleAtIndex:buttonIndex]);
}
```

Alert Operations

delegate property
title property
message property
visible property

– addButtonWithTitle:
numberOfButtons property
– buttonTitleAtIndex:
cancelButtonIndex property
firstOtherButtonIndex property

– show

– dismissWithClickedButtonIndex:animated:

UIAlertViewDelegate Protocol

Responding to Actions

- alertView:clickedButtonAtIndex:

Customizing Behavior

- willPresentAlertView:
- didPresentAlertView:
- alertView:willDismissWithButtonIndex:
- alertView:didDismissWithButtonIndex:

Canceling

- alertViewCancel:

Alert & Background process

iOS 3

When application is terminated

alerts are closed automatically

iOS 4

When application move to background or terminated

alerts not closed automatically

Your code has to decide when to close alert

But how?

App Delegate Methods

```
- (BOOL)application:(UIApplication *)application
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions

- (void)applicationWillResignActive:(UIApplication *)application

- (void)applicationDidBecomeActive:(UIApplication *)application

- (void)applicationWillTerminate:(UIApplication *)application

- (void)applicationDidReceiveMemoryWarning:(UIApplication *)application
```

Simplistic Solution

AlertActionSheetAppDelegate

```
- (void)applicationWillResignActive:(UIApplication *)application {
    [viewController movingToBackground];
}
```

AlertActionSheetViewController

```
- (void) movingToBackground {
    [sampleAlert
        dismissWithClickedButtonIndex:[sampleAlert cancelButtonIndex]
        animated:YES];
}
```

Designing An Alert

Single line titles

Use sentence fragments

Avoid single word titles

Error, Warning

Avoid using

"you", "your", "me", "my"

Designing An Alert - title

Use Title-Style Capitalization And No Punctuation

- Title is a sentence fragment
- Title is single sentence & not question

Use sentence-style capitalization ending with "?" for question

- Use question if it replaces additional message

Use sentence-style capitalization & ending punctuation for complete sentences.

- Should be rare
- Consider multi-sentence titles if it allows you to avoid adding a message

Designing An Alert - message

Designing An Alert - message

Designing An Alert - message

Avoid descriptions of which button to tap

If you must then

- Use "tap" not "touch", "click" or "choose"
- Don't enclose button title in quotes

Designing An Alert - message

Test in both orientations

Designing An Alert - Buttons

Prefer a two-button alert

Avoid single button alerts

Give people some control

Avoid alerts with 3+ buttons

Consider using Action sheet

Designing An Alert - Bu

Button colors - dark & light

One Button
light

Two Buttons
Left dark Right light

Designing An Alert - Buttons

If potentially risky action

Two Buttons

Action dark Cancel light

If benign action that people are likely to want

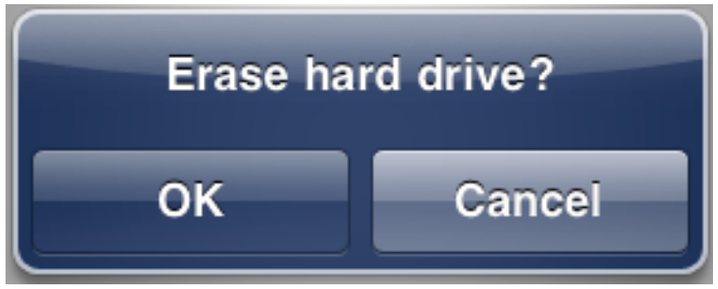
Two Buttons

Cancel dark Action light

Designing An Alert - Button titles

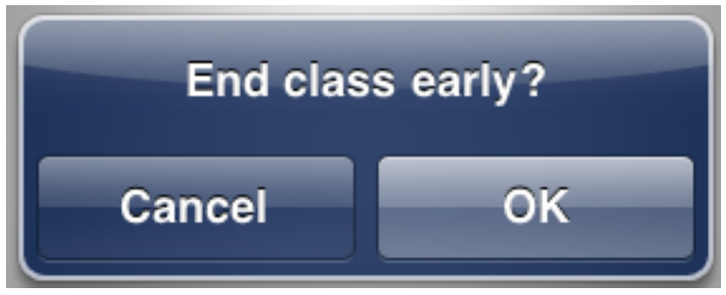
Prefer verbs and verb phrases

“Cancel,” “Allow,” “Reply,” or “Ignore”



Prefer "OK"

Avoid "Yes" & "No"



Avoid “you” “your” “me” “my”

iPhone Human Interface Guideline