

# WordCollage

## Lesson 4



### Description

Add two more buttons that change the background to different colors.

### Learning Outcomes

- Repeat adding buttons to the interface with Interface Builder, defining layout constraints, and establishing action connections to controller code.
- Analyze how Xcode indicates when interface controls are connected to controller methods.
- Describe why connections between interface controls must change when their corresponding controller method names change.
- Differentiate multiple ways of establishing and managing connections between interface controls and controller code.
- Distinguish different connections between individual interface elements and code.



### Vocabulary

Interface Builder	Assistant Editor	user interface
Button	constraint	controller
connection	implement	connection well
connection overlay	@IBAction	UIColor

## Materials

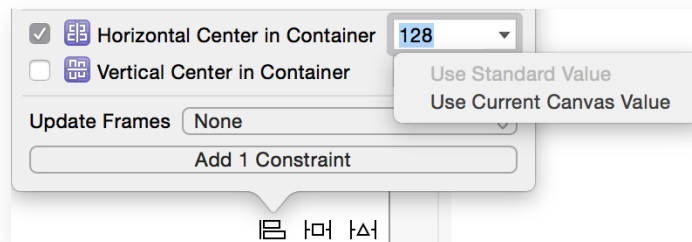
- **WordCollage Lesson 4** Xcode project

## Opening

How do you create two more buttons that change the background to different colors?

## Agenda

- Change the label of the existing Button contents to "Black."
- Using Interface Builder and the Object Library (⌘⇧L), add a Button to the bottom left of the interface, labeled "White."
- Using Interface Builder, Control-drag from the Button downward to the View, and select Bottom Space to Bottom Layout Guide to create a Vertical Space constraint.
- With the Button still selected, use the Align control and select Horizontal Center in Container using the Current Canvas Value to create a Center X Alignment constraint.



- Add another button, labeled "Magenta," to the bottom right of the interface, and add constraints similar to the previous Button.
- Using Interface Builder and the Assistant Editor (⌘⇧⌘⇧), establish connections between each button and two new controller methods,  
changeBackgroundColorToWhite: and changeBackgroundColorToMagenta:.

```
@IBAction func changeBackgroundColorToWhite(sender: UIButton) {  
}
```

```
@IBAction func changeBackgroundColorToMagenta(sender: UIButton) {  
}
```

- Implement the two methods.

```
@IBAction func changeBackgroundColorToWhite(sender: UIButton) {  
    view.backgroundColor = UIColor.whiteColor()  
}  
  
@IBAction func changeBackgroundColorToMagenta(sender: UIButton) {  
    view.backgroundColor = UIColor.magentaColor()  
}
```

- Rename `changeBackgroundColor:` to `changeBackgroundColorToBlack:`, and observe that the adjacent connection well appears hollow.
- Run the app (⌘R), tap the Black button, and witness the app crash. Stop the app (⌘.).
- Explain that the app crashed because Interface Builder still tries to connect the button to the `changeBackgroundColor:` method, which no longer exists.
- Using Interface Builder and the connection overlay, delete the old connection, establish a new connection to `changeBackgroundColorToBlack:`, and observe the connection well reappear.
- Run the app (⌘R), tap the buttons and witness the background color changing.

## Closing

Cool apps require lots more code. What should we learn about Swift so we can build more sophisticated, meaningful apps?

## Modifications and Extensions

- When tapped, change the button colors themselves in addition to the background color.
- Replace the three separate controller actions with a single `changeBackgroundColor:` method, connected to all three buttons. Implement the method such that it distinguishes between which button is tapped, and changes the background color accordingly.

## Resources

iOS Developer Program <https://developer.apple.com/programs/ios/>

Start Developing iOS Apps Today <https://developer.apple.com/library/ios/referencelibrary/GettingStarted/RoadMapiOS/>

iOS Technology Overview <https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/>

iOS App Programming Guide: About iOS App Programming <https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/Introduction/Introduction.html>

Xcode Overview: Build a User Interface [https://developer.apple.com/library/ios/documentation/ToolsLanguages/Conceptual/Xcode\\_Overview/edit\\_user\\_interface.html](https://developer.apple.com/library/ios/documentation/ToolsLanguages/Conceptual/Xcode_Overview/edit_user_interface.html)

Adding an Object to Your Interface [https://developer.apple.com/library/ios/recipes/xcode\\_help-IB\\_objects\\_media/Chapters/AddingObject.html](https://developer.apple.com/library/ios/recipes/xcode_help-IB_objects_media/Chapters/AddingObject.html)

Xcode Overview: Connect User Interface Objects to Code [https://developer.apple.com/library/ios/documentation/ToolsLanguages/Conceptual/Xcode\\_Overview/edit\\_user\\_interface.html#//apple\\_ref/doc/uid/TP40010215-CH6-SW3](https://developer.apple.com/library/ios/documentation/ToolsLanguages/Conceptual/Xcode_Overview/edit_user_interface.html#//apple_ref/doc/uid/TP40010215-CH6-SW3)

Cocoa Application Competencies for iOS: Target-Action <https://developer.apple.com/library/ios/documentation/General/Conceptual/Devpedia-CocoaApp/TargetAction.html>

Using Swift with Cocoa and Objective-C: Working with Outlets and Actions [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/BuildingCocoaApps/WritingSwiftClassesWithObjective-CBehavior.html#//apple\\_ref/doc/uid/TP40014216-CH5-XID\\_62](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/BuildingCocoaApps/WritingSwiftClassesWithObjective-CBehavior.html#//apple_ref/doc/uid/TP40014216-CH5-XID_62)

The Swift Programming Language: Attributes [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/Attributes.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Attributes.html)