Gesturizer
Lesson 4

Description

Use an additional closure to control the fading out of the gesture label.

Learning Outcomes

- Practice using Swift closure syntax.
- Discover how closures may call methods that pass additional closures as arguments.
- Recognize the Swift syntax for closures that receive an argument when invoked.
- Discover how the underscore character can represent an ignored parameter name.
- Combine different forms of Swift closure expression syntax to express a problem solution.

Vocabulary

alpha transparency	`UIView`	closure
type annotation	parameter	return type
`Void`	closure body	underscore

Materials

- **Gesturizer Lesson 4** Xcode project
- **Closures** presentation

Opening

How can we ensure that the label fades both in and out for each gesture?

Agenda

- Discuss the existing implementation of `showGestureName:`, and how, during the first invocation, the label starts as transparent and animates to full opacity; yet for subsequent invocations the label "animates," but from full opacity to full opacity.
- Using the Xcode Documentation and API Reference (⇧⌘0), explore the `UIView` class reference and the `animateWithDuration:animations:completion:` class method.
- Update the implementation of `showGestureName:`.

```
func showGestureName(name: String) {
   self.gestureName.text = name
   UIView.animateWithDuration(1.0,
      animations: { self.gestureName.alpha = 1.0 },
      completion: { (finished) -> Void in
         self.gestureName.alpha = 0
   })
}
```

- Explain how the method `animateWithDuration:animations:completion:` accepts a second closure, which will execute when the `animations:` closure is complete.
- Explain how the second `completion:` closure expression is different from the first, in that it will receive a single `Bool` argument when invoked.
- Present the concept of closures (if necessary).
- Discuss how the `completion:` closure expression declares an explicit `Void` return type and a `finished` parameter that the closure body does not use.
- Modify the implementation of the `completion:` closure expression.

```
UIView.animateWithDuration(1.0,
   animations: { self.gestureName.alpha = 1.0 },
   completion: { _ in self.gestureName.alpha = 0 })
```

- Explain the underscore, in this context, as a Swift convention for an ignored parameter name.
- Run the app (⌘R), tap the screen, and observe the label fade in and then quickly disappear.
- Discuss how the `completion:` closure sets the label alpha attribute to `0`, which causes the label to immediately disappear.
- Discuss what may be necessary in the `completion:` closure expression is another `UIView` animation.

- Update the implementation of `showGestureName:`.

```
func showGestureName(name: String) {
   self.gestureName.text = gestureName
   UIView.animateWithDuration(1.0,
      animations: { self.gestureName.alpha = 1.0 },
      completion: { _ in
         UIView.animateWithDuration(1.0) { self.gestureName.alpha = 0 }
   })
}
```

- Discuss how the first closure expression, passed as the `animations:` argument, handles the fading in of the label; and how the animation within the `completion:` closure expression handles the fading out of the label.
- Run the app (⌘R), tap the screen, and observe the label fade in and out.

Closing

What happens when we double-tap slowly? Do you see the conflict in our animation? Why do you think it is occurring?

Modifications and Extensions

- Explore the label's `transform` attribute, the `CGAffineTransformMakeScale` structure, and the `CGAffineTransformIdentity` constant. Implement a transformation effect that makes the label appear to fade into or out from the screen, and continues to work with subsequent gestures.

Resources

UIKit User Interface Catalog: About Views https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/UIKitUICatalog/index.html

UIView Class Reference https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIView_Class/index.html

The Swift Programming Language: Closures https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Closures.html