iPhone Programming
Textfields and Special-Purpose Views

Tim Gegg-Harrison

Protocols

- Protocols
  - Enable you to declare an interface, which a class implements, without providing any default implementation of that interface
  - Solution to the “diamond problem” with multiple inheritance (extending 2 different classes which each have a distinct implementation of the same method)
    - Same as an interface in Java
  - Classes implement (or conform to) protocols
    - A class can conform to multiple protocols

Protocols …

- Protocols
  - Declaring a protocol:

```
protocol SomeProtocol {
   // protocol definition goes here
}
```

Example:

```
protocol DiceGameDelegate {
  func gameDidStart(game: DiceGame)
  func game(game: DiceGame,
       didStartNewTurnWithDiceRoll diceRoll: Int)
  func gameDidEnd(game: DiceGame)
}
```

Textfields

- The `UITextField` class encapsulates a text-editing control that allows the user to enter a small amount of information.
- Important properties of text fields:

`text`
- Obtain and set the text that is displayed in the text field

`textAlignment`
- Alignment of the text that is displayed in the text field

`textColor`
- Color of the text that is displayed in the text field

`background`
- Image that represents the background of the text field

Textfields …

- The `UITextField` class encapsulates a text-editing control that allows the user to enter a small amount of information.
- Important properties of text fields:

`clearButtonMode`

- Manages the appearance of the clear button of the text field

`UITextFieldViewMode.never`
– clear button never appears

`UITextFieldViewMode.whileEditing`
– clear button appears only when user is editing the text field

`UITextFieldViewMode.unlessEditing`
– clear button appears only when user is not editing the text

Textfields …

- The `UITextField` class encapsulates a text-editing control that allows the user to enter a small amount of information.
- Important properties of text fields:

`borderStyle`

- Manages the appearance of the border style of the text field

`UITextBorderStyle.none`
– default

`UITextBorderStyle.line`

`UITextBorderStyle.bezel`

`UITextBorderStyle.roundedRect`

Textfields …

- The `UITextField` class encapsulates a text-editing control that allows the user to enter a small amount of information.
- Important properties of text fields:

  `delegate`
  - Delegate of the text field `UITextFieldDelegate`

  `disabledBackground`
  - Background image used when the text field is disabled (if non-`nil`)

  `editing`
  - Read-only property indicating if the text field is in edit mode

  `font`
  - Font of the text that is displayed in the text field

Textfields …

- The `UITextField` class conforms to the `UITextInputTraits` protocol.
- Important properties of the `UITextInputTraits` protocol:

  `keyboardType`
  - Controls the style of keyboard associated with the text field

    `UIKeyboardType.default`
    - Default keyboard

    `UIKeyboardType.asciiCapable`
    - Keyboard that displays standard ASCII characters

    `UIKeyboardType.numbersAndPunctuation`
    - Keyboard with numbers and punctuation

Textfields …

- The `UITextField` class conforms to the `UITextInputTraits` protocol.
- Important properties of the `UITextInputTraits` protocol:

`keyboardType`

- Controls the style of keyboard associated with the text field

`UIKeyboardType.numberPad`

- Numeric keyboard suitable for PIN entry

`UIKeyboardType.phonePad`

- Keyboard designed for entering phone numbers

`UIKeyboardType.namePhonePad`

- Keyboard designed for entering a person's name and

Textfields …

- The `UITextField` class conforms to the `UITextInputTraits` protocol.
- Important properties of the `UITextInputTraits` protocol:

`keyboardType`

- Controls the style of keyboard associated with the text field

`UIKeyboardType.decimalPad`

- Keyboard with numbers and a decimal point (iOS 4.1)

`UIKeyboardType.twitter`

- Keyboard optimized for Twitter entry – easy access to # and @
  (iOS 5.0)

Textfields …

- The `UITextField` class conforms to the `UITextInputTraits` protocol.
- Important properties of the `UITextInputTraits` protocol:

`secureTextEntry`

- Used to signal that text entered should be hidden (characters replaced by asterisks)

`returnKeyType`

- Used to define the title for the return key

```
UIReturnKey.default, UIReturnKey.go,
UIReturnKey.google, UIReturnKey.join,
UIReturnKey.next, UIReturnKey.route,
UIReturnKey.search, UIReturnKey.send,
UIReturnKey.yahoo, UIReturnKey.done,
UIReturnKey.emergencyCall
```

Textfields …

- The `UITextField` class conforms to the `UITextInputTraits` protocol.
- Important properties of the `UITextInputTraits` protocol:

`keyboardAppearance`

- Dark vs. Light look of keyboard (Default is same as Light)

```
UIKeyboardAppearance.default
UIKeyboardAppearance.dark
UIKeyboardAppearance.light
```

`enableReturnKeyAutomatically`

- If the value is `true` then the keyboard's return key is disabled until the user enters some text

Textfields …

- The `UITextField` class conforms to the `UITextInputTraits` protocol.
- Important properties of the `UITextInputTraits` protocol:

`autocorrectionType`

- Used to manage the autocorrection of the user's input

```
UITextAutocorrectionType.default
UITextAutocorrectionType.no
UITextAutocorrectionType.yes
```

`autoCapitalizationType`

- Determines when shift key is automatically pressed to produce capital letters

```
UITextAutoCapitalizationType.none
UITextAutoCapitalizationType.words
UITextAutoCapitalizationType.sentences
UITextAutoCapitalizationType.allCharacters
```

Textfields …

- The `UITextField` class conforms to the `UITextInputTraits` protocol.
- Important properties of the `UITextInputTraits` protocol:

`spellCheckingType`

- Used to manage the spell checking of the user's input (iOS 5.0)

```
UITextSpellCheckingType.default
UITextSpellCheckingType.no
UITextSpellCheckingType.yes
```

Textfields …

- The `UITextField` class uses the `UITextFieldDelegate` protocol for communicating with the delegate class.
- Important tasks of the `UITextFieldDelegate` protocol:

`textFieldShouldBeginEditing:`
  - Asks the delegate if editing should begin in the text field

`textFieldDidBeginEditing:`
  - Tells the delegate that editing began in the text field

`textFieldShouldEndEditing:`
  - Asks the delegate if editing should stop in the text field

`textFieldDidEndEditing:`
  - Tells the delegate that editing stopped in the text field

Textfields …

- The `UITextField` class uses the `UITextFieldDelegate` protocol for communicating with the delegate class.
- Important tasks of the `UITextFieldDelegate` protocol:

`textField:shouldChangeCharactersInRange:replacementString`
  - Asks the delegate if specified text should be changed

`textFieldShouldClear:`
  - Asks the delegate if text field's current contents should be removed

`textFieldShouldReturn:`
  - Asks the delegate if the text field should process the pressing of the return button

Textfields …

- Creating a text field:

```
let textField1: Textfield
textField1.frame = CGRect(x: centerX-100, y: centerY-125,
width: 200, height: 50)
textField1.textColor = UIColor.black
textField1.font = UIFont.systemFont(ofSize: 17.0)
textField1.placeholder = "<enter text>"
textField1.backgroundColor = UIColor.white
textField1.borderStyle = UITextBorderStyle.bezel
textField1.keyboardType = UIKeyboardType.default
textField1.returnKeyType = UIReturnKeyType.done
textField1.clearButtonMode = UITextFieldViewMode.always
textField1.delegate = self
self.view.addSubview(textField1)
```

Textfields …

- Implementation of the `textFieldShouldReturn:` method:

```
func textFieldShouldReturn(_ textField: UITextField) -> Bool {
    if myTextField == textField {
      if myTextField.text == "Hide" {
        myTextField.resignFirstResponder()      // hide KB
      }
    }
    return true
}
```

Special-Purpose Views

- **`UIPickerView`**
  - Demo:
    - **`PickerView`** (delegate + dataSource)
- **`UIProgressView`**
- **`UIActivityIndicatorView`**
  - Demo:
    - **`ProgressView`**
- **`UIScrollView`** (delegate)
  - Demo:
    - **`ScrollView`**

Special-Purpose Views …

- **`UITextView`**
  - implements the behavior for a scrollable, multiline text region
  - supports the display of text using a custom font, color, and alignment and also supports text editing
  - typically used to display multiple lines of text, such as when displaying the body of a large text document
  - does not support multiple styles for text
    - font, color, and text alignment attributes you specify always apply to the entire contents of the text view
    - To display more complex styling in your application, you need to use a **`UIWebView`** object
      - allows presenting rich content to the user