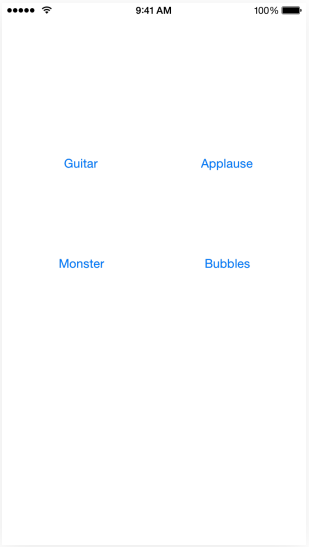
NoiseMaker
Lesson 5

Description

Add a `NoiseMaker` model to the project to encapsulate all audio playback.



Learning Outcomes

- Analyze a code listing and decide what should be encapsulated in a model.
- Assess the use of access control, such as private, when declaring properties.
- Practice writing class definitions, declaring properties and defining methods.
- Distinguish between constants and variables, and between optional and non-optional types.

Vocabulary

Model-View-Controller	model	class
property	`private`	method
optional	forced unwrapping	constant
default value		

Materials

- **NoiseMaker Lesson 5** Xcode project

Opening

What should our model be, and what should it encapsulate?

Agenda

- Add a new `NoiseMaker` class (⌘N) to the project to encapsulate the model.
- Copy the four controller `AVAudioPlayer?` properties into the `NoiseMaker` model, mark them as `private`, and import the `AVFoundation` framework.

```
import AVFoundation

class NoiseMaker {

   private let guitarPlayer: AVAudioPlayer
   private let applausePlayer: AVAudioPlayer
   private let monsterPlayer: AVAudioPlayer
   private let bubblesPlayer: AVAudioPlayer

}
```

- Discuss the decision to make each property private, since the model will provide methods to play each sound, rather than providing access to each `AVAudioPlayer` directly.
- Add four new methods to the `NoiseMaker` class for playing each sound. Copy code that exists in the controller actions into each respective model method.

```
func playGuitarSound() {
   let url = NSBundle.mainBundle().URLForResource("guitar",
      withExtension: "wav")
   guitarPlayer = AVAudioPlayer(contentsOfURL: url, error: nil)
   guitarPlayer!.play()
}

func playApplauseSound() {
   let url = NSBundle.mainBundle().URLForResource("applause",
      withExtension: "wav")
   applausePlayer = AVAudioPlayer(contentsOfURL: url, error: nil)
   applausePlayer!.play()
}
```

```
func playMonsterSound() {
   let url = NSBundle.mainBundle().URLForResource("monster",
      withExtension: "wav")
   monsterPlayer = AVAudioPlayer(contentsOfURL: url, error: nil)
   monsterPlayer!.play()
}

func playBubblesSound() {
   let url = NSBundle.mainBundle().URLForResource("bubbles",
      withExtension: "wav")
   bubblesPlayer = AVAudioPlayer(contentsOfURL: url, error: nil)
   bubblesPlayer!.play()
}
```

- In the `ViewController` class, remove the the `AVFoundation` framework import.
- In the controller, replace the four `AVAudioPlayer?` properties with a single property for a `NoiseMaker` model, including a default value.

```
let noiseMaker = NoiseMaker()
```

- Discuss how the `noiseMaker` property is a constant, and how, unlike the individual `AudioPlayer?` properties, is not an optional since a default value is provided.
- Starting with the `playGuitar:` method, replace the audio playback contents of each controller action with a method call using the model.

```
@IBAction func playGuitar(sender: UIButton) {
   noiseMaker.playGuitarSound()
}
...
@IBAction func playBubbles(sender: UIButton) {
   noiseMaker.playBubblesSound()
}
```

- Run the app (⌘R), tap each button, and observe that the app still plays each sound correctly.
- Discuss how the controller now has no concern with the details of playing sounds. Discuss how the controller merely handles touch events from the view, and communicates with the `NoiseMaker` model.

Closing

We now have a model, but do you feel like we've just "hidden" the repetitive code inside the model? How can we improve this?

Modifications And Extensions

- Initialize the `NoiseMaker` with a dictionary of sound filenames and `AVAudioPlayer` objects. Refactor each of the controller methods to use a single model `play:` method that receives a `String` representing the name of a sound to play.

Resources

Cocoa Core Competencies: Model-View-Controller https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

Start Developing iOS Apps Today: Design Patterns https://developer.apple.com/library/ios/referencelibrary/GettingStarted/RoadMapiOS/DesignPatterns.html

Cocoa Core Competencies: Model Object https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/ModelObject.html

The Swift Programming Language: Classes and Structures https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/ClassesAndStructures.html

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

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

AV Foundation Programming Guide https://developer.apple.com/library/ios/documentation/AudioVideo/Conceptual/AVFoundationPG/Articles/00_Introduction.html

NSBundle Class Reference https://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSBundle_Class/index.html

NSURL Class Reference https://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSURL_Class/index.html

AVAudioPlayer Class Reference https://developer.apple.com/library/ios/documentation/AVFoundation/Reference/AVAudioPlayerClassReference/index.html

The Swift Programming Language: If Statements and Forced Unwrapping https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html#//apple_ref/doc/uid/TP40014097-CH5-ID332