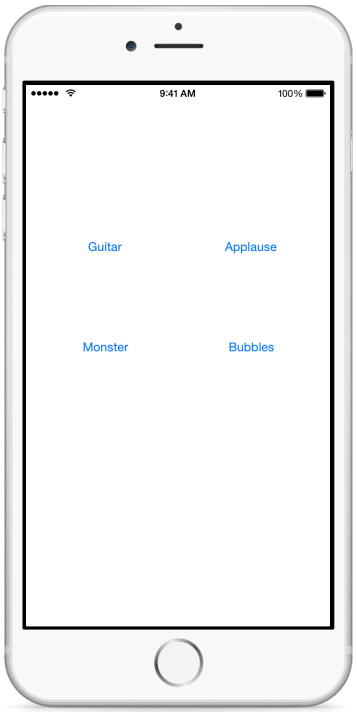
NoiseMaker
Lesson 3

Description

Implement the actions for each button, playing each of the four sounds.



Learning Outcomes

- Practice implementing controller methods to carry out interface behavior.
- Explain what URLs are, and discover how the `NSURL` class represents a URL.
- Discover how application bundles represent the files associated with an app, and how the `NSBundle` class abstracts the app bundle.
- Practice using the `AVAudioPlayer` API to play a sound file.
- Observe Swift error handling syntax and optional binding, and optional chaining.

Vocabulary

URL	path	`NSURL`
app bundle	`NSBundle`	`AVAudioPlayer`
optional binding	`try?`	optional chaining

Materials

- **NoiseMaker Lesson 3** Xcode project

Opening

How can we use an `AVAudioPlayer` to play a sound when a button is tapped?

Agenda

- Implement the controller method `playGuitar:`.

```
@IBAction func playGuitar(sender: UIButton) {
    if let url = NSBundle.mainBundle().URLForResource("guitar",
        withExtension: "wav") {
        player = try? AVAudioPlayer(contentsOfURL: url)
        player?.play()
    }
}
```

- Run the app (⌘R), tap the Guitar button, and listen to the guitar sound.
- Using the Xcode Documentation and API Reference (⇧⌘0), explore the `NSURL` and `NSBundle` class references.
- Explain that an `NSURL` represents a path to a particular file or even a network resource.
- Explain that an `NSBundle` represents a location of files and resources, and how the `mainBundle` method returns the bundle representing the location of the app files and resources.
- Explain that, because `URLForResource:withExtension:` returns an `NSURL?`, optional binding is necessary to safely unwrap the `NSURL` before it is passed to the `AVAudioPlayer` initializer.
- Using the Xcode Documentation and API Reference (⇧⌘0), explore the `AVAudioPlayer init(contentsOfURL:)` initializer, and observe that the initializer is marked with `throws`.
- Explain that `try?` is used with the `AVAudioPlayer` initializer to convert a possible error to an optional, and that optional chaining is used to safely call the `play` method.
- Using the Xcode Documentation and API Reference (⇧⌘0), search the documentation for `initWithContentsOfURL:`, and observe how many classes use this URL idiom.
- Implement the `playApplause:`, `playMonster:`, and `playBubbles:` methods.

```
@IBAction func playApplause(sender: UIButton) {
   if let url = NSBundle.mainBundle().URLForResource("applause",
      withExtension: "wav") {
      player = try? AVAudioPlayer(contentsOfURL: url)
      player?.play()
   }
}

@IBAction func playMonster(sender: UIButton) {
   if let url = NSBundle.mainBundle().URLForResource("monster",
      withExtension: "wav") {
      player = try? AVAudioPlayer(contentsOfURL: url)
      player?.play()
   }
}

@IBAction func playBubbles(sender: UIButton) {
   if let url = NSBundle.mainBundle().URLForResource("bubbles",
      withExtension: "wav") {
      player = try? AVAudioPlayer(contentsOfURL: url)
      player?.play()
   }
}
```

- Run the app (⌘R), tap on each button, and listen to each sound.
- Tap on each button quickly, observe how the currently playing sound stops, and how the new sound immediately begins playing.

Closing

Why does one sound stop when another begins playing?

Modifications and Extensions

- Using the OS X Finder, navigate to **~/Library/Developer/CoreSimulator/Devices/[DEVICE_ID]/data/Containers/Data/Application/[APP_ID]**. Ctrl-click **NoiseMaker.app**, and select **Show Package Contents** from the menu. Describe what you see in relation to application bundles.
- Bind the four buttons to just one controller method that plays a different sound according to which button is tapped.
- When using one controller method, design an approach to playing a different audio file based on closures instead of an `if` or `switch` statement.
- Explore Swift error handling, and use a `do-catch` statement when instantiating the `AVAudioPlayer`.

Resources

Bundle Programming Guide: Accessing a Bundle's Contents http://developer.apple.com/library/ios/documentation/CoreFoundation/Conceptual/CFBundles/AccessingaBundlesContents/AccessingaBundlesContents.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 Optional Binding https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html#//apple_ref/doc/uid/TP40014097-CH5-ID333

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

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