Mr. Brautigam's Zoo

Overview

- Create a multi-screen app. Use segues to navigate between scenes.
- Connect view controllers and code using the identity inspector.

Contents

Learning Outcomes

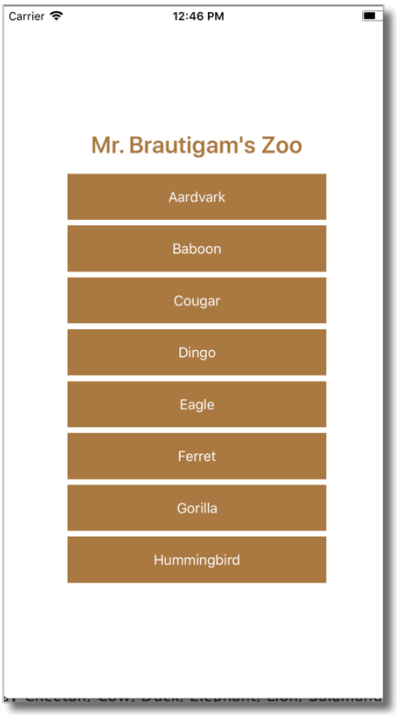
- Practice adding a new class to an Xcode project.
- Discover how interface components in Interface Builder can be bound to specific classes.
- Practice establishing **outlet** connections between a view and controller, and declaring properties.
- Practice establishing an **action** connection from a view to a controller method.
- Implementing button behaviors to add additional features to an app.
- Practice adding a new view controller to a project.
- Practice adding storyboard segues to connect view controllers to each other.
- Practice sharing data between view controllers.

Vocabulary

- UIView
- UITouch
- UIImageView
- UIImage
- Aspect Fit
- Aspect Fill
- subclass
- superclass
- override
- Cocoa Touch Class
- Swift File
- event
- Stack View
- button
- label
- outlet connection
- property
- optional
- optional binding
- if let
- action
- segue
- unwind segue
- layer
- sublayer
- gradients
- HSL color model
- Navigation Controller
- Navigation Bar
- Refactoring
- Build Phases
- Bundle Resources

Part 1: Create the first screen

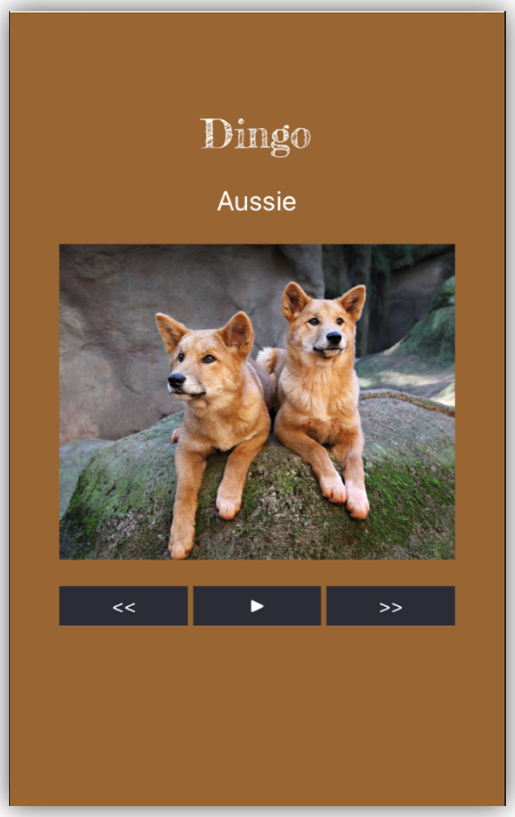
Objective: Add buttons to the view and put animal names on them.



1. Create a new app.
    a. iOS app, single page application,
    b. Call it something like "Animals" or "Zoo."
    c. Language is Swift, etc., all the usual stuff.
2. Put a **button** on the storyboard.
    a. Put a background color on the button.
3. Duplicate and place 6 more buttons so there is a total of 7 buttons.
4. Put a label above the buttons and change the text to "Family Photos" or something similar or funny.
5. Put the buttons and the label in a **stack view.**
    a. Use **constraints** to center the stack view vertically and horizontally.
    b. Use constraints to specify the width of the stack view. You can either specify a specific width, or you can use the I-beams to move the edges closer to the screen edges.
6. The buttons should say Aardvark, Baboon, Cougar, Dingo, Eagle, Ferret, Gorilla.

Part 2: Add a second scene with animal images

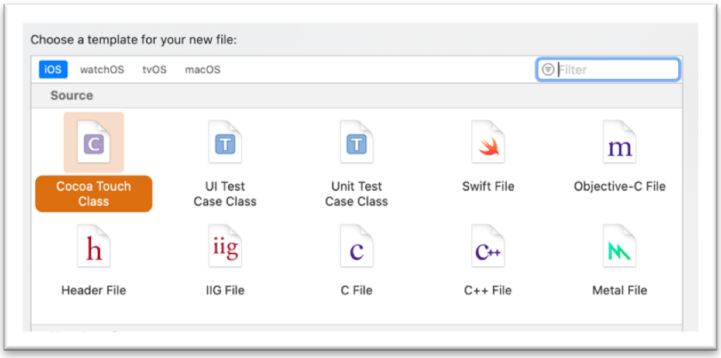
Objective: Add a new view controller that shows detail about one animal.



1. Drag the 7 images into the **Assets** catalog of your app.
2. Drag a **new View Controller** onto the screen.
3. Drag a new **UIImage** into the new View Controller.
    a. Set the image mode to "Aspect Fit" (or maybe “Aspect Fill”).
    b. Set the image to "aardvark".
4. Drag a new **Label** onto the new View Controller.
    a. Make the font bigger, 30 points or so, and label it "Aardvark".
5. Drag a new **Button** onto the new View Controller.
6. Label the button "Home".
7. Put the label, the image, and the button into a **stack view.**
8. Maybe put a width or height or aspect ratio on the image or the stack view. (We will talk about this together.)
9. Put a **background color** on the view.
10. Create segues from the various Animal buttons to their animal views.
    a. Control-drag from the Aardvark button to the view controller that has the image. Select the “show” or “show detail” segue.
    b. Control-drag from the other animal buttons to the view controller that has the image.
11. Create a segue from the Back button back to the main view that has the animal names.

Part 3: Add code for the second scene

Objective: Add a new view controller class and connect it to the second scene.



1. Make a new view controller:
    a. New File
    b. iOS Source
    c. Cocoa Touch Class
    d. Class = AnimalViewController
    e. Subclass of = UIViewController
    f. Language = Swift
    g. Next →
    h. Animals folder
    i. Create/OK
2. Connect the view controller with an Aardvark to Class = **AnimalViewController**
    a. Use the Identity Inspector ( ).
    b. Use the Custom Class dropdown menu.
3. Assistant Editor: Add IBOutlets for the label and the image
    a. animalCaption
    b. animalImage
4. Add code to **ViewController.swift** to tell AnimalViewController which animal to display. To start, just type "prepare" and let Xcode type the rest. Add the code in green.

```
override func prepareForSegue(
    segue: UIStoryboardSegue, sender: Any?) {
    let destination = segue.destination as! AnimalViewController
    let button = sender as! UIButton
    destination.animalCaption.text = button.title(for: .normal)
    if let title = button.title(for: .normal) {
        destination.animalImage.image = UIImage(
            named: title.lowercased()
    }
}
```

5. The above code may crash. We will fix the problems in Part 4.

Part 4: Make the segue work properly

Objective: Fix the crash and provide a method of communication between view controllers using the segue mechanism.

The app crashes because the destination image and caption are not set up until its view controller's **viewDidLoad** method triggers, which hasn't happened yet at the time we activate the **prepareForSegue** method in the source view controller. So we will create some temporary variables to hold the information we need. Then we'll set up the image and caption in the viewDidLoad method when they are set up and ready to go.

1. Add a new variable in the **AnimalViewController** file

```
var whichAnimal = “”
```

2. Add code in the ViewController.swift file, **prepareForSegue** function:

```
destination.whichAnimal = button.title(for: .normal)
```

3. Add some code to the **viewDidLoad** method in the **AnimalViewController.swift** file:

```
override func viewDidLoad() {
    super.viewDidLoad()

    // Do any additional setup after loading the view.
    animalCaption.text = whichAnimal
    animalImage.image = UIImage(named: whichAnimal.lowercased())
}
```

4. Add background colors in viewDidLoad after setting the caption and image. (You could also use a **switch** statement here.)

```
if whichAnimal == “Aardvark” {
    view.backgroundColor = .blue
}
else if whichAnimal == “Baboon” {
    view.backgroundColor = .green
}
etc … for all possible images
```

Part 5: Create a database of animals

Objective: Create a searchable database of animals with various data to display.

1. Make a new file to put the data model into.
    a. File -> New
    b. iOS Source
    c. Swift File
    d. AnimalData.swift
    e. Save it in the Animals folder
    f. Drag it into the Animals folder in the outline if necessary
2. Create a struct called **Animal**
    a. Add a String property called name
    b. Add a UIColor property called bgcolor
3. Add **import UIKit** near the top of the file
4. Add an array called **AnimalData** and start initializing it with data, like this:

```
var animalData = [
    Animal (
        name : "Aardvark",
        bgcolor : .brown,
    Animal ( …
    etc. … for all the other animals
```

5. Add a method for finding a particular animal in the list.

```
func findAnimal (_ name : String) -> Int {
    for i in 0..<animalData.count {
        if animalData[i].name == name {
            return i
        }
    }
    return 0
}
```

6. In the **AnimalViewController** file, remove the long series of if…else statements, and change the viewDidLoad method like this:

```
let n = findAnimal (whichAnimal)
view.backgroundColor = animalData[n].bgcolor
```

Part 6: Add custom caption and colors

Objective: Add a subtitle to each image and set custom text colors.

1. In the **Animal** data structure, add two new properties: subtitle (a String) and textColor (a UIColor).
2. In the **AnimalData** array, add subtitle and textColor for each animal. For subtitles, give the animals goofy names. For textColor, you can use a light color if your background is dark. You can use a dark color if your background is light. It's OK to use white and black for text.
3. In the scene with the image, add a **label** in the stack view, below the title, above the image.
    a. Set a font size for the subtitle label, smaller than the caption.
    b. Add an **outlet** for the subtitle.
4. Add code to set the text of the subtitle to the selected animal's subtitle

```
subtitle.text = animalData[n].subtitle
```

5. Add code to set the color of the caption and subtitle text to the animal's text color.

```
subtitle.textColor = animalData[n].textColor
animalName.textColor = animalData[n].textColor
```

Part 7: Add an unwind segue to the home screen

Objective: On the animal scene and add an unwind segue to the Home button to go back to the home screen in the correct manner.

1. Delete the segue from the Home button back to the home screen.
    a. Just select it and delete it.
2. Add segue unwind code to the home screen view controller.
    a. You can name the function anything you want, but it must be labeled **@IBAction** and it must have one parameter, of type **UIStoryBoardSegue**.
    b. This is one of the few times you'll type @IBAction into your code instead of control-dragging.
    a. We won't need any code inside this function.

```
@IBAction func unwindToHomeScreen(sender: UIStoryboardSegue) {
}
```

3. In the Interface Builder, control-drag from the Home button to the **Exit icon** at the top of the view controller.
    a. In the menu, select **Unwind**, then select the name of the function you created above.
4. After you've done this, when you run the app, the animal view scene will disappear off the bottom of the screen, instead of the home screen coming back onto the screen.

Part 8: Add rounded corners and gradients to the buttons

Objective: Add rounded corners and gradients to the buttons on the home screen and the animal view screen.

1. Code to create gradients looks something like this. (This makes a medium green gradient.) This code does not have to be in the loop.

```
let topcolor = UIColor(red: 0, green: 192, blue: 0, alpha: 1).cgColor
let bottomColor = UIColor(red: 0, green: 128, blue: 0, alpha: 1).cgColor
```

2. You could also use the **HSL color model** to create your gradients. The following code makes a light blue gradient. 240 degrees is blue; low saturation and high brightness make it a light blue.

```
let topColor = UIColor(
   hue: 240.0/360.0, saturation: 0.2, brightness: 1.0, alpha: 1).cgColor
let bottomColor = UIColor(
   hue: 240.0/360.0, saturation: 0.2, brightness: 0.8, alpha: 1).cgColor
```

3. Add an outlet collection for the buttons on the home screen. You'll loop through the buttons to add styles like this:

```
for button in buttons {
}
```

4. Code to add rounded corners and gradients looks something like this.

```
// you may need this if you use a dark background color
button.tintColor = .white

// create and add gradient layer
let gl = CAGradientLayer()
gl.color = [ topcolor, bottomColor ]
gl.locations = [ 0, 1 ]
gl.frame.size = button.frame.size
button.layer.insertSublayer(gl, at: 0)
button.layer.sublayers![0].cornerRadius = 25.0 // corner on gradient

// rounded corner for the button
button.layer.cornerRadius = 25.0 // corner on button
```

5. If you want, you could add similar code to the AnimalViewController. There is only one button there now (Home), but we will add more buttons later.

Part 9: Add sound

Objective: Add animal sounds and play them in response to a button press.

1. Unzip the sound file and add the sounds into the files section of the **Project Navigator,** in the yellow group (folder).
    a. When asked, tell the dialog box to "copy" the files if necessary.
    b. Note: although sounds are considered assets in media, we don't add them into the **Assets.xcassets** folder. That folder is only for image assets.
2. Add the **SimpleSound.swift** file into your project. Add it into the files section of the Project Navigator, along with the other Swift files.
3. Add a property for the audio file name into your **Animal** structure.

```
var audioName : String
```

4. Add all the audio file names into the **Animal** entries in the **AnimalData** array.
5. **Embed** the **Home** button on the **AnimalViewController** scene in a **stack view.**
    a. Change the stack view orientation to Horizontal if necessary.
    b. Distribute = fill equally.
    c. Spacing = whatever you want, but at least a couple of pixels.
6. Add a new button into the stack view.
    a. Change the button's title to use an emoji for sound, such as a speaker.
    b. Control-drag to add an action method for the sound button.
7. Put a variable to store the sound at the top of the view controller code.

```
var sound : SimpleSound?
```

8. In the action method, use this code:

```
sound = SimpleSound(named: animalData[which].audioName)
sound.play()
```

Part 10: Add previous and next navigation buttons

Objective: Add previous and next buttons so we don't have to go back to the home screen before seeing a new animal.

1. In the stack view with the Home button and the audio play button, add two more buttons: **Previous** and **Next**. You might find emojis or symbols to put on these icons, or you might just use < and >.
2. Add **action methods** for the previous and next buttons.
3. Move applicable code from the **viewDidLoad** method into a new **loadAnimal()** method.
4. The code to go to the **previous** animal might look something like this:

```
var which = findAnimal(whichAnimal)
which -= 1
if which < 0 {
    which = animalData.count-1
}
whichAnimal = animalData[which].name
loadAnimal()
```

5. The code to go to the **next** animal might look something like this:

```
var which = findAnimal(whichAnimal)
which += 1
which %= animalData.count
whichAnimal = animalData[which].name
loadAnimal()
```

Note: some years, we use a dictionary instead of a simple array. In that case, the code for previous and next is more complicated.

Part 11: Use Google Fonts

Objective: Add Google Fonts to give the app a special look.

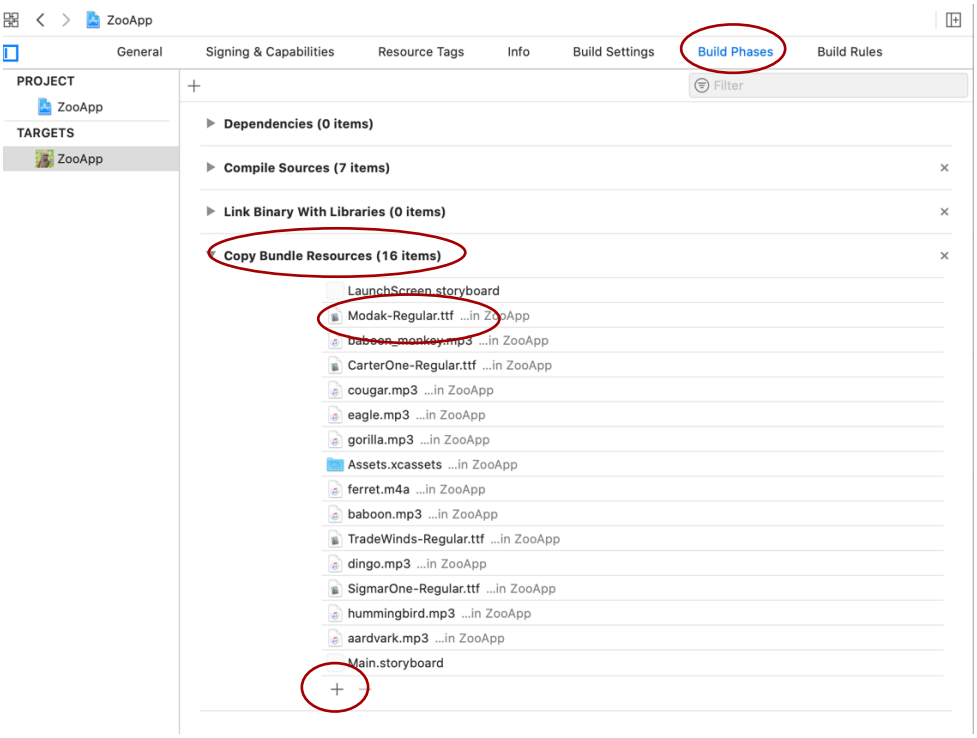
Note: This part was problematic in the AM class, and we just skipped it in the PM class. However, some students did have success, and I found a couple of web sites with hints and code that might be helpful.

In recent versions of Xcode, you can just download the font from Google, add it into your files, use it in your storyboard. But there is one more special step: you must add it to the resources that would get copied to the iPhone or other device (including the Simulator).

1. Find a font you like on Google Fonts.
2. Download the font.
3. Drag the font into the files section of the Project Navigator, along with the Swift files and audio files.
4. Using Interface Builder, select the button or label where you want to use a special font, then select your font from the list of custom fonts. Xcode may not recognize your font right away. In that case, you may try:
    a. Closing your project and opening it again.
    b. Quitting Xcode and starting it up again.
    c. Using some of the tricks below.
5. Check out these sites for tips:
    a. https://codewithchris.com/common-mistakes-with-adding-custom-fonts-to-your-ios-app/
    b. https://www.simpleswiftguide.com/how-to-use-custom-fonts-in-swift-ios-app-using-swiftui/
    c. I found this line of code for Coding with Chris to be helpful. Note: you have to plug in the correct name of your font (without ttf or otf file extension) and the size you want. You'll also need an outlet to the label or button where you want to set the font.

```
label.font = UIFont(name: "QuicksandDash-Regular", size: 35)
```

6. Add the font into the list of Bundle Resources. See the example below.
    a. Select the blue project icon at the top of the Project Navigator.
    b. Select the Build Phases tab.
    c. Click the + button at the bottom of the list.
    d. Choose your font file and add it.



Part 12: Add a Navigation Controller

Objective: Add a Navigation Controller for navigating back to the home screen "the Apple way."

1. Using Interface Builder, select the first screen (the buttons with animal names) by selecting the bar at the top of the scene.
2. Select **Embed** and from the menu, choose **Navigation Controller.**
    a. You'll see a new gray "phantom" view controller to the left of your first view controller. This is normal.
3. When you run the app and navigate to an animal using the buttons, the animal view screen will come in from the right instead of from the button. (If you chose show or show detail as your segue method.)
4. Your animal view scene will have a Back button at the top, in the navigation bar.
    a. You'll no longer need the back button at the bottom of this scene. You could delete it if you want to.
5. You can put names on your screens in the navigation controller.
6. Using code, you could put the animal names themselves in the navigation bar. The code might look something like this:

```
navigationItem.title = animalData[which].animalCaption
```

Part 13: Add a new animal to the list

Objective: Add the hummingbird.

1. Add a button on the home page that says Hummingbird.
2. If you made an outlet collection for your buttons, so you can add rounded corners and gradients to your buttons, make sure to add the new button to the outlet collection.
3. Add a hummingbird item to end of the animal database. You'll need to fill in all the fields, including nickname/subtitle, background color, and text color. A background color of green or blue works well to start.
4. After you add the new animal to your database, it should automatically get added into the navigation.

Part 14: Finishing touches

Objective: Make the app more polished.

1. Check both portrait and landscape orientation on a smaller phone, such as iPhone SE.
2. If the buttons don't all fit on the screen in landscape orientation, consider tightening the spacing.
3. You probably need to tighten the spacing only in landscape orientation, so you might consider using a size class to change the spacing only in hC (compact height, any width).
4. Another option might be to have a different set of stack views that arranges the buttons in two columns on an iPhone with compact height.
5. Check the view on an iPad.
6. Make sure the buttons on your animal view controller screen (with the photo) are visible. The best way to do this is to put a background color on them. If you have to, you could add a field to the Animal structure to indicate the button colors, so you always have a contrasting button color.
7. Consider adding the rounded corners and gradients to the buttons on the animal view controller screen if you added them on the first screen.
8. Don't forget to check the animal view scene on a small iPhone in landscape orientation. Tighten up the spacing on the stack view if you need to.
9. Add icons to the previous, next, play, and home buttons if you want to.
10. If your navigation controller works, you can remove the Home button, because it is now redundant. You don't actually have to delete it. You can just un-check the "installed" checkbox at the bottom of the Attributes Inspector.