armadsen Fix typo 1e4480c on Aug 30, 2018

1 contributor

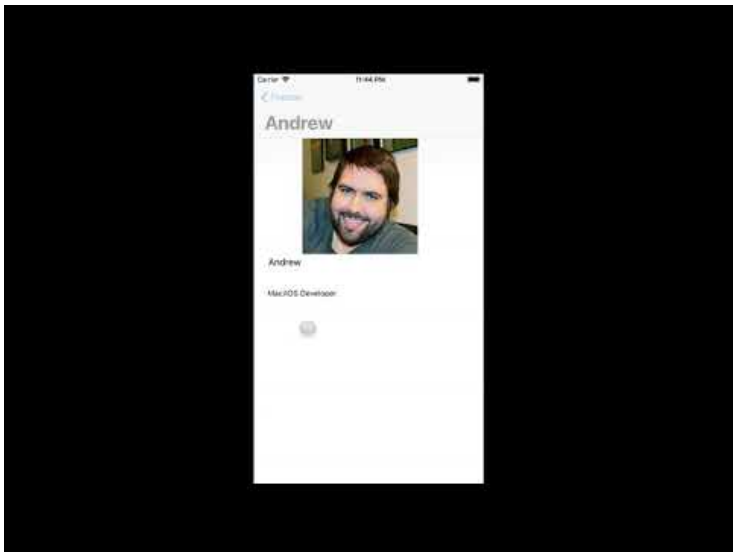
64 lines (40 sloc) 4.74 KB

Friends

For this project, you'll build an app that shows a table view of friends including their photo and name. When a row is tapped, a detail view should be shown. The detail view should include a bigger photo, name, and a short bio. Most importantly, implement a custom view controller transition so that the table view cell's image and name appear to smoothly transition to their new sizes and positions on the detail view. Watch the video below to see what your finished application should look like.

After completing the lesson material and this project, you should be able to:

- create custom transition animations using the view controller transition animation API
- create interactive transition animations



Part One - Basic App

Start by implementing the basic app, without the custom transition. It's a simple master/detail interface like you've implemented many times before. For model data, include one or more `Friend` instances just so you have content for the table view.

Part Two - Implement Animation Scaffolding

There are a number of ways to approach this custom transition, but perhaps the most straightforward is to to implement a `UINavigationControllerDelegate` that provides a custom animator to the navigation controller.

Create NavigationControllerDelegate

1. Create a class called `NavigationControllerDelegate`. It must inherit from `NSObject`, and should conform to the `UINavigationControllerDelegate` protocol.
2. Add `navigationController(_ navigationController:, animationControllerFor:, from:, to:)`. Leave it empty for now.
3. Create a `sourceCell` property so the navigation controller delegate can be given a reference to the table view cell that initiated the transition.

Create an Animator Class

1. Create a class called `ImageTransitionAnimator`. It must inherit from `NSObject` and should conform to the `UIViewControllerAnimatedTransitioning` protocol.
2. Implement `transitionDuration(using:)`. Return a sane animation duration. 0.5 is a good choice.
3. Add `animateTransition(using:)`. This is the main animation function for your custom transition. Leave it empty for now.
4. Add properties for the views that the animator will need to know about. These are the source and destination photo views, and the source and destination name labels.

Hook Everything Up

You'll need to provide the navigation controller with a delegate. You'll also need to set the animation up when the table view cell -> detail view segue is triggered.

1. In your table view controller's `viewDidLoad()`, set the `navigationController`'s delegate to an instance of `NavigationControllerDelegate`.
2. Implement `prepare(for segue:)`. Get the tapped row, and use it to give the navigation controller delegate the source table view cell.
3. In your `NavigationControllerDelegate`'s `navigationController(_ navigationController:, animationControllerFor:, from:, to:)` method, create and configure an `Animator` object, and return it. The animator will need to have its source and destination imageView and label properties set. **Note: You must keep a strong reference to the animator object in your navigation controller delegate. Use a property to hold it instead of creating a temporary instance and referencing it solely in this method.**

Part Three - Implement the Animation

You're ready now to implement the actual animation.

1. Implement `Animator.animateTransition(using:)`
2. Get the start and end frames for the image view and label.
3. Create a temporary image view and label to be used for the animation and add them to the transitionContext's `containerView`.
4. Set up any properties that should be set up before beginning the animation. For example, hide the destination views by setting their alpha to 0.0.
5. Use `UIView.animate()` to animate the transition.
6. In the animation's completion closure, (re)set the correct properties for the destination views, and remove the temporary animation views. Call `transitionContext.completeTransition()`.

Part Four - Testing

1. Run the project and make sure everything works.
2. If anything doesn't work the way the video shows, spend time debugging it and fixing the problem.
3. As always, if you need help, follow the 20-minute rule, then ask your PM.

Go Farther

If you finish early or want to push yourself, here are a few additional features you can implement:

- Make it so the back action can be perform using a swipe guesture like a standard navigation controller. The animation should be interactive and should track the user's finger. Hint: You'll need to use a `UIPanGuestureRecognizer`.
- Make the table view cell's image view rounded. Animate the corner radius changing to make it square on the detail view.