

View Controller Transitions

Challenge Instructions

Copyright © 2014 Razeware LLC.

All rights reserved. No part of this book or corresponding materials (such as text, images, or source code) may be reproduced or distributed by any means without prior written per- mission of the copyright owner.

This book and all corresponding materials (such as source code) are provided on an "as is" basis, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, and non-infringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software.

All trademarks and registered trademarks appearing in this book are the property of their respective owners.

## Challenge: On The Right Track!

You may have noticed that whenever you dismiss a photo, it leaves a big white blank whole where the photo originated from. Also, there really isn’t any nice dismiss animation to compliment your new presentation animation. You’re going to add all of this right now!

Open **FullscreenDismissAnimator.swift** and find the **animateTransition:** function. This is the exact same function that you were working on in FullscreenPresentationAnimator except you are going to be adding the dismiss animation now.

Make your implementation of animateTransition: look like the following:

func animateTransition(ctx: UIViewControllerContextTransitioning) {

let to = ctx.viewControllerForKey(UITransitionContextToViewControllerKey)! as PhotosController

let from = ctx.viewControllerForKey(UITransitionContextFromViewControllerKey)! as PhotoDetailController

let duration = transitionDuration(ctx)

}

Just like in your FullscreenPresentationAnimator class, you need to get all of the configuration and state for the current transition. Notice how the to and from controllers are flipped. This is because you are coming **from** the PhotoDetailController and moving **to** the PhotosController.

Beneath the line where you get the duration of the animation, add in the following:

UIView.animateWithDuration(duration,

delay: 0,

usingSpringWithDamping: 0.7,

initialSpringVelocity: 0.2,

options: .BeginFromCurrentState,

animations: {

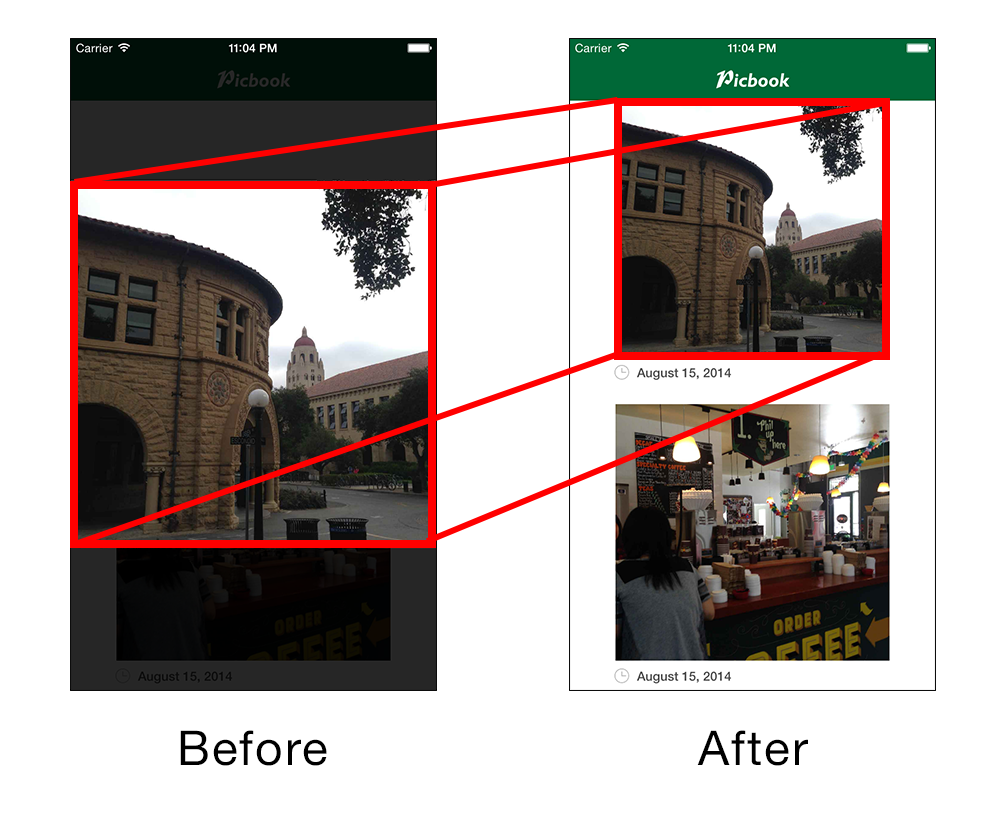
// TODO...

}, completion: { finished in

// TODO...

})

Just like in the Lab, you need to animate the state from one controller to another and then perform any cleanup. Try filling in the TODO statements by using the diagram below, and if you get stuck, refer back to the work you did in FullscreenPresentationAnimator.



In the animation completion block, do some additional cleanup to make sure that the from controller’s view is no longer loitering and to make sure that the image is visible when the animation finishes.

UIView.animateWithDuration(duration,

// ...

}, completion: { finished in

// your code...

to.selectedView?.hidden = false

from.view.removeFromSuperview()

})

The last thing that you need to do is make sure that you have a nice, smooth background fade along with the photo dismiss animation. At the end of **animateTransition:** add the following:

UIView.animateWithDuration(duration \* 0.2, animations: {

from.view.backgroundColor = UIColor.clearColor()

})

**Build and run** your app, open a photo, then tap or drag and release to dismiss it to observe your fancy new transition animation! It should look something like the sequence below.

