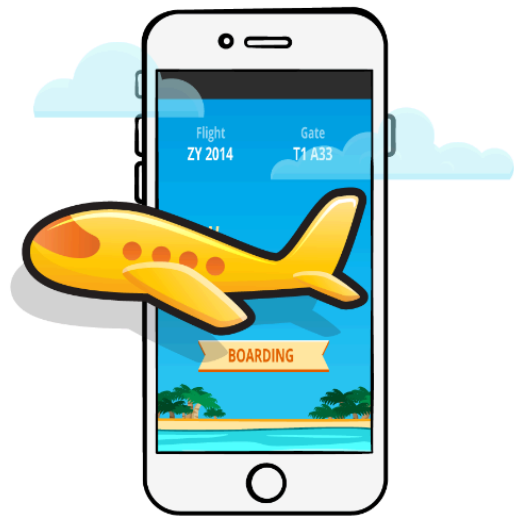


INTERMEDIATE

ios ANIMATIONS



HANDS-ON CHALLENGES

Intermediate iOS Animations

Catie & Jessy Catterwaul

Copyright ©2017 Razeware LLC.

Notice of Rights

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 permission of the copyright owner.

Notice of Liability

This challenge 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 noninfringement. 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 of other dealing in the software.

Trademarks

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

Table of Contents: Overview

Challenge 3: Intermediate Property Animators	5
--	---

Table of Contents: Extended

Challenge 3: Intermediate Property Animators	5
Dismiss the menu	5

Challenge 3: Intermediate Property Animators

By Catie & Jessy Catterwaul

Dismiss the menu

You have a cool peek and pop animation, but no way to dismiss it once complete! To do this, create yet another static method in **AnimatorFactory.swift** below the methods you just created:

```
//TODO: Dismiss the menu
static func reset(
    frame: CGRect,
    view: UIViewEffectView,
    blurView: UIViewEffectView
) -> UIViewPropertyAnimator {
}
```

This method takes in a CGRect for the view's initial frame, the main view you want to animate, as well as a blurView.

Inside of the method, create an animator that will reset everything you've just animated in the video back to its initial state:

```
return UIViewPropertyAnimator(
    duration: 0.5,
    dampingRatio: 0.7
) {
    view.transform = .identity
    view.frame = frame
    view.contentView.alpha = 0

    blurView.effect = nil
}
```

This is another spring animation. In it, you undo any transform changes with `.identity`, and reset the frame. You also make the view's label invisible once more. Finally, you animate the blur away by setting the effect to `nil`.

Now you just need to call this method and then do a little clean-up work. Back in **LockScreenViewController**, add the following to the `dismissMenu()` method:

```
let reset = AnimatorFactory.reset(  
    frame: startFrame!,  
    view: previewEffectView,  
    blurView: blurView  
)  
  
reset.startAnimation()
```

With the `reset` method, you use the `startFrame` that you stored earlier to reset `previewEffectView`'s frame. You also pass in `blurView` to reverse its effects. Finally, you start the animator.

If you scroll down a bit, you'll see that `dismissMenu` is called from a tap gesture recognizer attached to `previewEffectView`. If you build and run, tap on an icon, and then tap on the little effect view, the peek and pop animation will successfully be reversed! But things may not work as you expect if you try again.

You don't want the views you just animated hanging around causing trouble like this, so you have a bit of clean-up work left to do. Because you're using a property animator, you can just add a completion handler right here! Just before you start the animator, add:

```
reset.addCompletion { _ in  
    self.previewEffectView.removeFromSuperview()  
    self.previewView?.removeFromSuperview()  
}
```

You remove both `previewEffectView` and `previewView` from the screen, and you're done! Build and run to test it out again.