

Intermediate iOS Animation

Hands-on Challenges

Intermediate iOS Animation Hands-On Challenges

Copyright © 2016 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 permission 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 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 or other dealings in the software.

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



Challenge A: More complex delegate animations

In this challenge you are going to add a completely new animation to the project and add a neat effect to enhance the visual appeal of the app. You are going to animate those clouds in the background and make them cross the screen and re-appear from the opposite side continuously.

First let's create a method in `viewController`, which takes any layer and animates it towards the right edge of the screen.

```
func animateCloud(cloudLayer: CALayer) {  
    //animate clouds  
    let cloudSpeed = 30.0 / Double(view.frame.size.width)  
    let duration = NSTimeInterval(view.frame.size.width -  
        cloudLayer.frame.origin.x) * cloudSpeed  
}
```

`animateCloud` takes a layer parameter and, after it calculates the average speed needed to cross the screen in 30 seconds, finds what duration does the layer need to reach the right edge from its current position.

Once you have the duration you can create the animation as usual:

```
let cloudMove = CABasicAnimation(keyPath: "position.x")  
cloudMove.duration = duration  
cloudMove.fromValue = cloudLayer.frame.origin.x  
cloudMove.toValue = view.frame.size.width +  
    cloudLayer.frame.size.width  
cloudMove.delegate = self  
cloudMove.timingFunction = CAMediaTimingFunction(name:  
    kCAMediaTimingFunctionLinear)
```

You set the from and to values, the duration and easing, and additionally you set the view controller as the animation delegate.

To be able to tell the cloud animation apart from the text fields animations you will set "name" and "layer" keys on the animation (just like you learned from the video lesson):

```
cloudMove.setValue("cloud", forKey: "name")
```



```
cloudMove.setValue(cloudLayer, forKey: "layer")
```

Finally you can add the animation to the layer and that's a wrap for that method:

```
cloudLayer.addAnimation(cloudMove, forKey: nil)
```

Scroll up to `presentationAnimations()` and add at the bottom:

```
animateCloud(cloud1.layer)
animateCloud(cloud2.layer)
animateCloud(cloud3.layer)
animateCloud(cloud4.layer)
```

This should get the clouds running across the screen :]

However, at present, once the clouds reach the right edge of the screen they remain there. You can "restart" them across the screen by calling `animateCloud` from the `animationDidStop` delegate method.

Scroll to `animationDidStop` and right after the last closing curly bracket inside the method add an `else` branch to the `if`:

```
else if name == "cloud" {
    layer.frame.origin.x = -layer.frame.size.width
    delay(seconds: 0.1, completion: {
        self.animateCloud(layer)
    })
}
```

You check if the currently completed animation has a name equal to "cloud" and if so – you move the cloud to the left side of the screen and after a little delay you call `animateCloud()` on it.

And that's a wrap! Run the project and observe the clouds crossing the sky and re-appearing from the opposite edge again and again.

