

CALayers

Hands-on Challenges

CALayers Hands-On Challenges

Copyright © 2015 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.

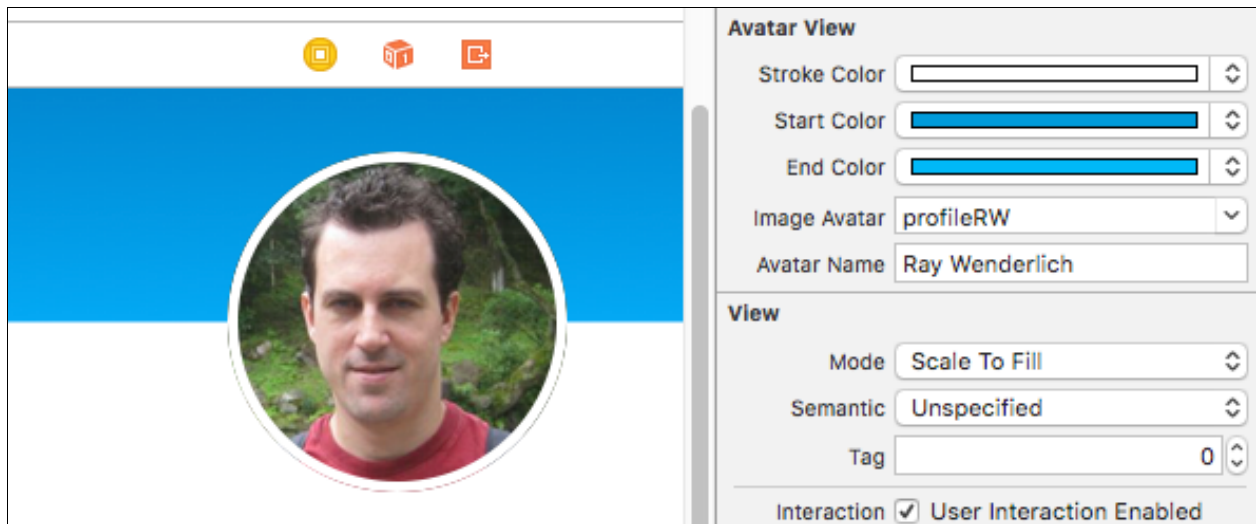
This book and all corresponding materials (such as source code) are provided on an "as is" basis, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, and noninfringement. In no event shall the authors or copyright 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 B: Configurable Gradient

Currently, the gradient is hard-coded to black and white. Wouldn't it be great if you could dynamically change those colors right from the Storyboard editor?



Thanks to a feature introduced in iOS 8 you can; and it's pretty easy. Let's see how it works.

Open **AvatarView.swift**. You'll notice that the class is marked with an `@IBDesignable` keyword:

```
@IBDesignable
class AvatarView: UIView {
```

This is why you are able to preview the control in the Storyboard editor. When `@IBDesignable` is set on a view, the Storyboard editor will attempt to render the view. It will call `prepareForInterfaceBuilder()` on startup, and it will call `layoutSubviews()` when the bounds change.

When you have an `@IBDesignable` view, you can easily set some properties as editable in the storyboard editor; just put the `@IBInspectable` tag beforehand. A few caveats:



- Only certain types are editable: Int, CGFloat, Double, String, Bool, CGRect, CGSize, CGPoint, UIImage, and UIColor.
- You must manually set the datatype for the property – you can't rely on implicit datatypes. In other words, this will **not** work (implicit typing):

```
@IBInspectable var strokeColor = UIColor.blackColor()
```

But this will work (explicit typing):

```
@IBInspectable var strokeColor: UIColor = UIColor.blackColor()
```

- Often you will want to set an observer when your property changes so you can reconfigure your layer properties appropriately.

Let's put this all together. Replace the definitions of `strokeColor`, `startColor`, and `endColor` with the following:

```
@IBInspectable var strokeColor: UIColor = UIColor.blackColor() {
    didSet {
        configure()
    }
}

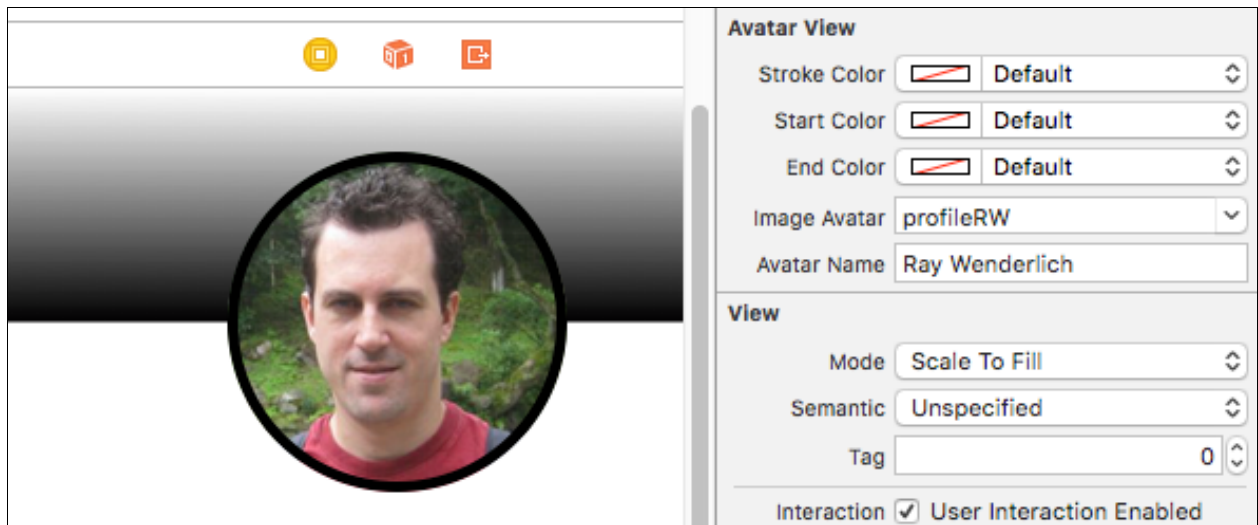
@IBInspectable var startColor: UIColor = UIColor.whiteColor() {
    didSet {
        configure()
    }
}

@IBInspectable var endColor: UIColor = UIColor.blackColor() {
    didSet {
        configure()
    }
}
```

This makes each property a `var` (rather than `let`) and calls `configure()` whenever the property changes so the layer properties can be updated appropriately.

Then open **Main.storyboard**, click your avatar view, and in the Attributes Inspector you'll see new options to select the colors:

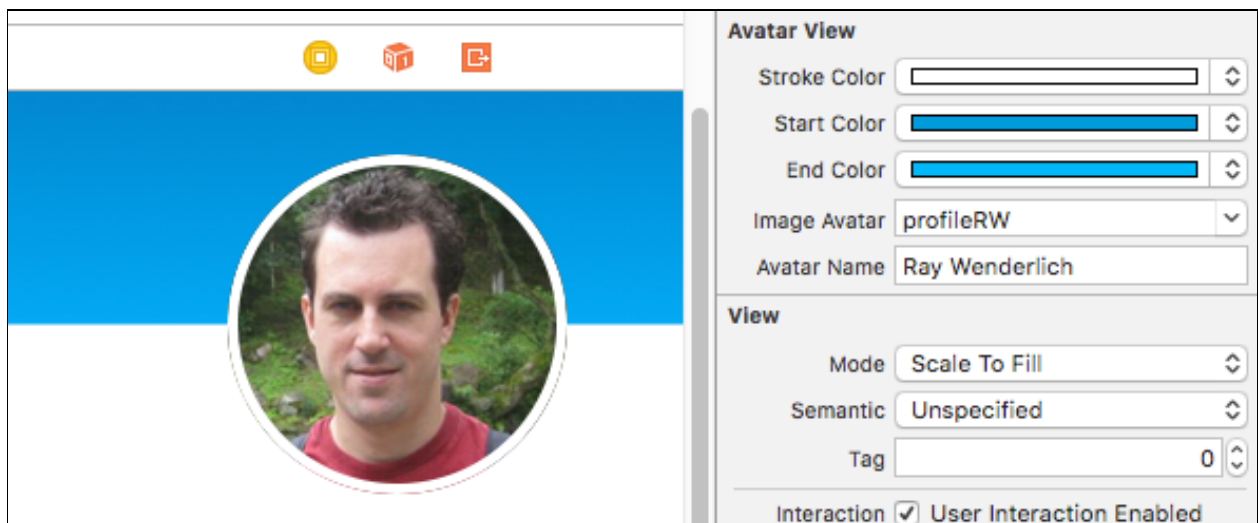




Use the controls to set the following colors:

- **Stroke Color:** #FFFFFF
- **Start Color:** #0288D1
- **End Color:** #03A9F4

You should then see the following:



Congrats! This makes it super handy to iterate on color choices.

