

CUSTOM CONTROLS IN iOS



HANDS-ON CHALLENGES

Custom Controls in iOS

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.

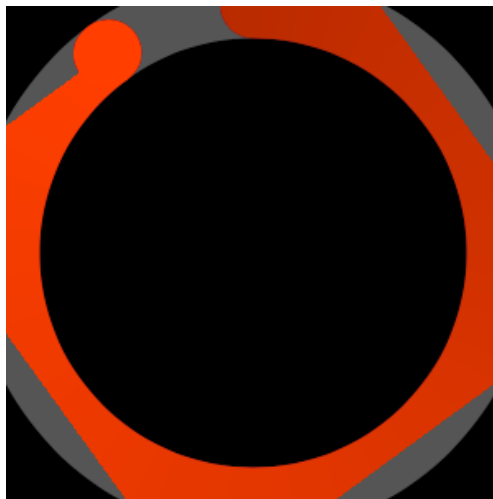
Trademarks

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

Challenge #5: Drawing with Layers

By Catie & Jessy Catterwaul

Despite now being able to change `RingLayer.value` willy-nilly, and have the ring visibly update accordingly, setting the `ringBackgroundColor` and `ringColor` properties has no effect, and while the `ringWidth` property certainly yields the ability to do...*something*...



...it's not what you might expect from *setting the ring's width*!

```
// Nope, still gray and red.  
ringLayer.ringBackgroundColor = Color.pink  
ringLayer.ringColor = Color.green  
  
// Ring by Picasso™  
ringLayer.ringWidth = 1
```

As you did in the demo, with `value`, you'll be able to use `didSet` observers on the problematic properties, to achieve the desired results. Working out precisely what needs to change, when each property changes, can lead to much more efficient code than simply redrawing the entire custom control when any of its properties change.

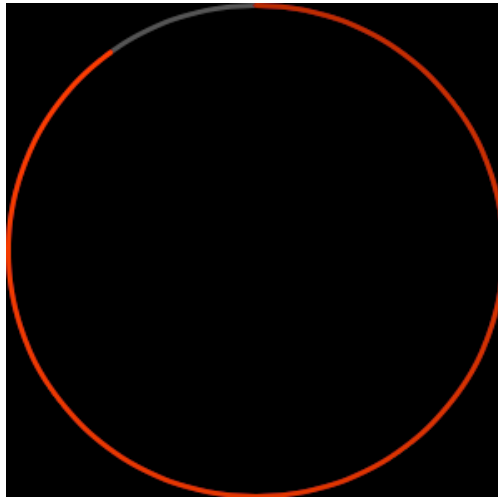
ringWidth

Three CALayers need to have their `lineWidth` properties match `ringWidth`. Set each of them, and then call `preparePaths`.

```
public var ringWidth: CGFloat = 40.0 {
    didSet {
        for layer in [
            backgroundLayer,
            foregroundMask,
            ringTipLayer
        ] {
            layer.lineWidth = ringWidth
        }
        preparePaths()
    }
}
```

Now you can change `ringLayer.ringWidth` as many times as you'd like!

```
ringLayer.ringWidth = 3
```



```
ringLayer.ringWidth = 100
```



ringColor

The two layers that are affected by `ringColor` are `gradientLayer` and `ringTipLayer`. Set the custom color property, for `gradientLayer`, and the `CAShapeLayer` property `strokeColor`, for `ringTipLayer`.

```
public var ringColor = UIColor.red.cgColor {  
    didSet {  
        gradientLayer.color = ringColor  
        ringTipLayer.strokeColor = ringColor  
    }  
}
```

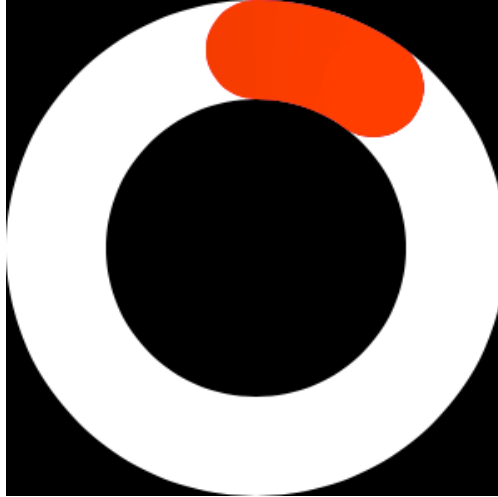


```
ringLayer.ringColor = Color.blue
```

ringBackgroundColor

Only `backgroundLayer.strokeColor` needs to change, when `ringBackgroundColor` changes.

```
public var ringBackgroundColor = UIColor.darkGray.cgColor {
    didSet {
        backgroundLayer.strokeColor = ringBackgroundColor
    }
}
```



```
ringLayer.value = 0.1
ringLayer.ringBackgroundColor = UIColor.white.cgColor
ringLayer.ringColor = UIColor.red.cgColor
```

CATransform3D Key Paths

As mentioned in the demo, it might not be obvious to you how this extension...

```
public extension CALayer {
    static let rotationKeyPath = "transform.rotation"
```

...could be used to rotate the two CALayers.

```
for layer: CALayer in [gradientLayer, ringTipLayer] {
    layer.setValue(
        getAngle(value: value),
        forKeyPath: CALayer.rotationKeyPath
    )
}
```

CALayer's has a transform property, but it's a CATransform3D, which doesn't have a rotation property that you can use in Swift. Access via key path is our only option. Read the section **Key Path Support for Structures** in Apple's [Key-Value Coding Extensions](#) guide, for more information.

Unfortunately, using a String directly in the extension is our best option. Swift 3's #keyPath syntax is unavailable for this API.