Intermediate iOS Animation

Hands-on Challenges

Intermediate iOS Animation Hands-On Challenges

Challenge A: More elaborate spring animation

With spring animations it is easy to get confused and restrict oneself to using them only on animations that have to do with position. But in reality there is no limit to what you can animate with a spring animation – you can achieve fun effects by animating transform.scale, borderWidth or the layer shadow.

In this challenge you are going to experiement with springy drop shadows. You will add a shadow to the currently selected text field. Let's start!

In **ViewController.swift** find the `textFieldDidBeginEditing()` method – this method gets called each time a text fields gets the focus. Great, half of the work is already done!

Add inside that method:

```
textField.layer.shadowColor = UIColor(white: 0.0, alpha:
0.5).CGColor
textField.layer.shadowOffset = CGSize(width: 5, height: 5)
textField.layer.masksToBounds = false
```

This code prepares the current text field for the animation. You set the shadow color to gray and give the shadow an offset of `5` points from the field itself.

Then you create a basic animation to fade in the shadow:

```
textField.layer.shadowOpacity = 1.0

let fadeIn = CABasicAnimation(keyPath: "shadowOpacity")
fadeIn.fromValue = 0
fadeIn.toValue = 1
fadeIn.duration = 1.0
textField.layer.addAnimation(fadeIn, forKey: nil)
```

And finally you can use a spring animation to animate the shadow offset:

```
textField.layer.shadowOffset = CGSize(width: 5, height: 7)
```

```
let shadowBounce = CASpringAnimation(keyPath: "shadowOffset")
shadowBounce.fromValue = NSValue(CGSize: CGSize.zero)
shadowBounce.toValue = NSValue(CGSize: CGSize(width: 5, height:
7))

shadowBounce.damping = 6.0
shadowBounce.duration = shadowBounce.settlingDuration

textField.layer.addAnimation(shadowBounce, forKey: nil)
```

You animate the `shadowOffset` property from zero (the zero size struct) to a `(5, 7)` offset from the field. This code already feels familiar because as you have certainly noticed the approach is the same whether you animate the layer's position or another property.

Run the app and click few times between the username and password fields.

You will see the shadow "jump" because of the shadow offset animation.

There's only one thing left to do – reset the shadow when the field loses focus. Add the proper delegate method to handle this event:

```
func textFieldDidEndEditing(textField: UITextField) {
    textField.layer.shadowOpacity = 0.0
}
```

That's beter :] As soon as you click the password field the username loses it shadow and vice versa.