Intermediate iOS Animation

Hands-on Challenges

Intermediate iOS Animation Hands-On Challenges

Challenge A: Grouping some more animations

In this easy and short challenge you are going to create another animation group for your project. Along the way you will learn how to override group properties for each particular animation.

Open **ViewController.swift** and scroll to `animateInfo()`. There are two animations that you create and run simultaneously so these are perfect candidates for building up an animation group.

Delete the whole block defining the two animations – better start from scratch. Remove the code:

```
let flyLeft = CABasicAnimation(keyPath: "position.x")
flyLeft.fromValue = info.layer.position.x + view.frame.size.width
flyLeft.toValue = info.layer.position.x
flyLeft.duration = 10.0
info.layer.addAnimation(flyLeft, forKey: nil)

let fadeIn = CABasicAnimation(keyPath: "opacity")
fadeIn.fromValue = 0.0
fadeIn.toValue = 1.0
fadeIn.duration = 5.0
info.layer.addAnimation(fadeIn, forKey: nil)
```

In its place create a new group:

```
let infoGroup = CAAnimationGroup()
infoGroup.beginTime = CACurrentMediaTime() + 0.5
infoGroup.duration = 10.0
infoGroup.fillMode = kCAFillModeBackwards
infoGroup.timingFunction = CAMediaTimingFunction(
    name: kCAMediaTimingFunctionEaseOut)
```

You give the whole group a delay and duration, as well as easing.

Next you will add the two animations to the group. First define the first animation like so:

```
let flyLeft = CABasicAnimation(keyPath: "position.x")
flyLeft.fromValue = info.layer.position.x + view.frame.size.width
```

```
flyLeft.toValue = info.layer.position.x
```

There's really nothing more to do here – all the values for missing properties like `duration` and `timingFunction` will be inherited from the group.

Now let create the second animation but make this one last only `5` seconds instead of the group defined duration of `10`:

```
let fadeIn = CABasicAnimation(keyPath: "opacity")
fadeIn.fromValue = 0.0
fadeIn.toValue = 1.0
fadeIn.duration = 5.0
```

You can see that setting the duration in both the concrete animation and the group works just fine – in this case the values you set directly on the animation itself have precedence. Thus the second animation will last only `5` seconds while the first one will go on for `10`.

Groups are a really flexible and powerful way to combine animations and produce more visually impressive effects!

The final step, of course, is to group the two animations and add them to the label's layer:

```
infoGroup.animations = [flyLeft, fadeIn]
info.layer.addAnimation(infoGroup, forKey: nil)
```

Run the app and notice how by the way the label is half-way in the screen it is already fully visible (unlike before when it took it the full `10`secs to fade in):