iOS Animation with Swift

Hands-On Challenges

iOS Animation with Swift Hands-On Challenges

Challenge 0: Finish up the pop soccer game

In this final challenge you will get to excercise your new Pop skills. To finish up the complete game you will make the ball bounce off the side edges of the screen.

To get started open **ViewController.swift**. Scroll to `pop_animationDidApply(animation:)` and at the bottom of the method body fetch the minimum and maximum x coordinates you will allow the ball to move to:

```
let minX = ball.frame.size.width/2
let maxX = view.frame.size.width - ball.frame.size.width/2
```

Since you are going to restrict the coordinates of the center of the ball view you allow for half of the ball width screen margin.

Now check whether the ball is outside of the bespoken margin and if so you will make the ball bounce. Add:

```
if ball.center.x < minX || ball.center.x > maxX {

}
```

First you will fetch the current velocity of the movement animation - add inside the `if`:

```
let velocityValue = animation.velocity as? NSValue
let velocity = (velocityValue?.CGPointValue())!
```

Since you are animating the ball position on screen, which is a `CGPoint` – its velocity is a `CGPoint` value as well. You get that velocity as a `CGPoint` by calling `CGPointValue()` on the `NSValue` object you get from the animation object.

This is very handy since you get separate values for the velocity along the x axis and the velocity on the y axis of the ball movement. To make the ball bounce off the left and right edges you will re-create the ball animation in its current state whenever it touches a screen edge. You will reverse the velocity on the x axis in the new animation and this will make the ball keep its vertical momentum, but bounce back horizontally.

Before you create the new animation remove the existing animations running on the ball. Add (still inside the `if`):

```
ball.pop_removeAllAnimations()
```

Next create a velocity equal to the current ball velocity but reversed horizontally:

```
let newVelocity = CGPoint(x: -velocity.x, y: velocity.y)
```

This will already make the ball reverse direction, but that's not quite enough – at the moment your replace the animation the ball is outside the allowed bounds. Just re-position it back inside the bounds – this won't make the ball jump since the change will be just few pixels:

```
let newX = min(max(minX, ball.center.x), maxX)
```

The combination of calling `mix(a:, b:)` and `min(a:, b:)` clamps the ball `x` coordinate between `minX` and `maxX`.

Finally re-create the decay animation but use the new velocity:

```
let newAnimation = POPDecayAnimation(propertyNamed:
kPOPViewCenter)
newAnimation.fromValue = NSValue(CGPoint: CGPoint(x: newX, y:
ball.center.y))
newAnimation.velocity = NSValue(CGPoint: newVelocity)
newAnimation.delegate = self
```

The new animation is ready to go, add:

```
ball.pop_addAnimation(newAnimation, forKey: "shot")
```

Believe it or not that's all you needed to do to create a cool bounce animation off the screen edges. If you want to extend the game just a bit more you can make the ball bounce off all of the screen edges.