Advanced watchOS

Hands-On Challenges

Beginning watchOS Hands-On Challenges

Challenge F: Uncomplicated Complication Data

Now that you have the complication set up, it's time to get some data in there!

In this short challenge, you'll access the weather data source and fill in some data to power the complication timeline entry.

Let's get started!

Weather data source

You have access to the data source in the regular app interface controller but not in the complication yet. Open **ComplicationController.swift** and add the following property to the class. This is the same code as in InterfaceController.swift, so you can copy-and-paste from there if you'd like:

```
lazy var dataSource: WeatherDataSource = {
  let defaultSystem =
   NSUserDefaults.standardUserDefaults().stringForKey(
   "MeasurementSystem") ?? "Metric"

  if defaultSystem == "Metric" {
    return WeatherDataSource(measurementSystem: .Metric)
  }
  return WeatherDataSource(measurementSystem: .USCustomary)
}()
```

This will create a lazy instance of the weather data source, using the user's stored setting for units.

In this case, the weather data is actually stored locally. Remember that for your own apps, it's important that your complication return data as soon as possible. If your watch app relies on the network or watch connectivity for all its data, try to have something cached locally for quick access from the complication.

Complication data

In the demo, the method you implemented was just for the template that's used when the user sets up the watch face.

Now that you have your data source and the complication template ready, there's one more method to implement — it's the one that provides the live complication data for display.

Add the following method to the class:

```
func getCurrentTimelineEntryForComplication(
 complication: CLKComplication,
 withHandler handler: (CLKComplicationTimelineEntry?) -> Void) {
  if complication.family == .UtilitarianSmall {
    let template = CLKComplicationTemplateUtilitarianSmallFlat()
    template.textProvider = CLKSimpleTextProvider(
     text: dataSource.currentWeather.temperatureString)
    template.imageProvider = CLKImageProvider(onePieceImage:
     UIImage(named:
     dataSource.currentWeather.weatherConditionImageName)!)

    handler(CLKComplicationTimelineEntry(date: NSDate(),
     complicationTemplate: template))
  }
}
```

This should look familiar; it's very similar to the code that sets up the initial complication template.

First up is the check for the complication family. In this case, you're only supporting the Utilitarian Small size, but you could add additional if statements for other sizes in the future.

Then as before, you get a template object and attach data providers to it. Here, the text will be the weather temperature from the data source, and the image will be the weather condition image.

The final step is to call the completion handler. For the template, all you needed to do was to pass in the template instance. Since this is live complication data, just a template isn't enough; you need to pass in a *timeline entry* object.

A timeline entry is just a wrapper around the template with an attached timestamp. Since it's current data, you can just pass in the current date and time with `NSDate()`.

In the next video on time travel, you'll pass in different date values here to go back and forward in time.