Gesture Recognizers

Events

There are 4 general types of UI events in iOS:

- **Touch** events: the most common
- **Motion** events
- **Remote-control** events: allow a responder object to receive commands from an external accessory or headset (usually to manage audio and video)
- **Press** events: represent interactions with a game controller, AppleTV remote, or other device that has physical buttons

Gestures

Gestures refer to touches and touch events.

- Central to the modern smart phone experience
- A core built-in capability in iOS

A *touch* is an instance of the user putting a finger on the screen.

The OS and the hardware work together to know when a finger touches the screen, where it is, when it moves, and when it is no longer touching the screen.

Its location at any point in time is reduced to a single appropriate point.

Gestures (cont.)

Why are they important?

- They allow us to interact more naturally and intuitively with the application
- It is a *significant* paradigm shift to how humans interact with computers: analogous to what happened when people were first provided GUIs to interact with computers

`UIResponder` Objects

`UIResponder` objects constitute the event-handling backbone of a `UIKit` app. As events occur, `UIKit` dispatches them to your app's responder objects for handling.

Key objects which are also responders include:

- `UIApplication`
- `UIViewController`
- All `UIView` objects (including `UIWindow`)

`UIResponder` Objects

To handle a specific type of event, a `UIResponder` object must override the corresponding methods. For example, to handle touch events, a responder might have to implement a method such as:

`touchesBegan(_:with:)` - tells the responder that one or more new touches occurred in a view or window

or

`touchesEnded(_:with:)` – tells the responder when one or more fingers are raised from a view or window

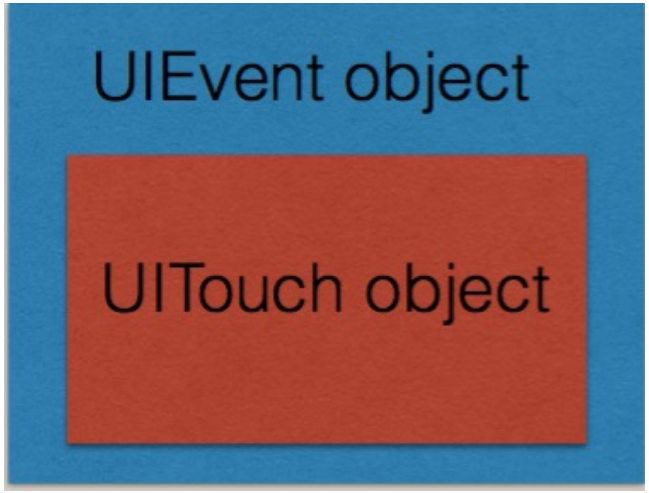
The responder would then use the event information provided by `UIKit` to track changes to those touches and to update the app's interface appropriately.

UIEvent and UITouch objects

A `UIView`, by virtue of being a `UIResponder`, is the central focus of touches.

It would make sense, then, if every touch were reported directly to the view in which it occurred; but this isn't the case since the OS doesn't see views, but only the entire app.

As such, a touch is represented as a `UITouch` object, which is bundled inside a `UIEvent` object - i.e. the *touch event*. The `UIEvent` object is then delivered to your application.



From a programmer's perspective, this means you respond to `UIEvent` objects, not `UITouch` objects.

Gesture Recognizers

Gesture recognizers are high-level mechanisms provided by iOS that takes care of the nitty-gritty of touch events, and makes it very easy to respond to a set of common touch events/sequences.

- They handle touches and movements of one or more fingers that happen on a specific area of the screen
- They are objects derived from the abstract *UIGestureRecognizer* class that are related to a view, and monitor for a predefined gesture made on that view
- There are some predefined subclasses which deal with specific (common) kinds of gestures
- They all perform an action once a valid gesture is detected.

Without gesture recognizers, you would be writing pages of code to handle what takes only a few lines of code *with* gesture recognizers.

Gesture Recognizers

You can set up gesture recognizers in IB or in code.

- A view can contain more than one gesture recognizer
- They are contained in a UIView property (an array) named gestureRecognizers

However, just one gesture can occur at any given point in time.

There are two types of gesture recognizers:

- Discrete: manage a single event; for example, touch to select an object
- Continuous: manage a series of events; for example, dragging an object on the screen

Gesture Recognizers

Predefined gesture recognizer classes:

- `UI`**`Tap`**`GestureRecognizer` (discrete)
- `UI`**`Swipe`**`GestureRecognizer` (discrete)
- `UI`**`Pan`**`GestureRecognizer` (continuous)
- `UI`**`Pinch`**`GestureRecognizer` (continuous)
- `UI`**`Rotation`**`GestureRecognizer` (continuous)
- `UI`**`LongPress`**`GestureRecognizer` (continuous)
- `UI`**`ScreenEdgePan`**`GestureRecognizer` (continuous)

Setting Up a Gesture Recognizer Using IB

- In IB, identify the object that you want to manipulate on the storyboard. Drag a Gesture Recognizer object **on top of the target object.**
- In the Swift file, write a function to handle the gesture.
- In IB, ctrl-drag the Gesture Recognizer object to the View Controller. Choose the name of the function you wrote.
- Click on the target object and go to the Attribute Inspector. Make sure "User Interface Enabled" is clicked on.

Setting Up a Gesture Recognizer Programmatically

- Create a Gesture Recognizer using one of the functions listed on the previous chart.

```
let tapRecognizer =
    UITapGestureRecognizer(target: self, action:
    #selector(handleTap(recognizer:)))
```

- Set up any properties for the Gesture Recognizer that you may want.
- Associate the Gesture Recognizer with the target object.

```
targetObject.addTapRecognizer(tapRecognizer)
```

- In the Swift file, write a function to handle the gesture.

```
@IBAction func handleTap(recognizer:
    UITapGestureRecognizer) {
        <code>
    }
```