iPhone Programming
Touches and Gestures

Tim Gegg-Harrison

Multitouch Interface

- Monitoring user touches
- Every application has a single **`UIApplication`** object for handling user touches
- When the user touches the screen, the system creates a **`UIEvent`** that contains a **`Set<UITouch>`** of **`UITouch`** objects (one for each finger) that is placed on the application's event queue
- The **`UIApplication`** object gets the event from the queue and sends it to the key window object
- The key window object determines which subview should receive the event and dispatches the event to it

Multitouch Interface …

- `UITouch` class
  - **`phase`**
    - current phase of the touch
      - `UITouchPhaseBegan`
      - `UITouchPhaseMoved`
      - `UITouchPhaseStationary`
      - `UITouchPhaseEnded`
      - `UITouchPhaseCancelled`
  - **`timestamp`**
    - time when touch changed its phase
  - **`tapcount`**
    - number of taps the user made when he/she touched the screen
  - **`locationInView`**
    - location of the touch in current view (`CGPoint`)
  - **`previousLocationInView`**
    - previous location of the touch in current view (`CGPoint`)

Multitouch Interface …

- `UIResponder` class
  - User interface objects (e.g., **`UIView`** objects) that receive touches are subclasses of the **`UIResponder`** class
    - `touchesBegan`
    - `touchesMoved`
    - `touchesEnded`

Multitouch Interface …

```
var offset: CGPoint

override func touchesBegan(_ touches: Set<UITouch>,
                           with event: UIEvent?) {
    let touch = touches.first!
    let touchedView: UIView = touch.view!
    offset = touch.location(in: touchedView)
    offset.x = touchedView.frame.size.width/2 - offset.x
    offset.y = touchedView.frame.size.height/2 - offset.y
    self.superview?.bringSubview(toFront: self)
}

override func touchesMoved(_ touches: Set<UITouch>,
                           with event: UIEvent?) {
    let touch = touches.first!
    let touchedView: UIView = touch.view!
    var location: CGPoint =
              touch.location(in: touchedView.superview)
    location.x += offset.x
    location.y += offset.y
    touchedView.center = location
}
```

Gestures

- Applications for iOS are driven largely through events generated when users touch buttons, toolbars, table-view rows and other objects in an application's user interface.
- The classes of the UIKit framework provide default event-handling behavior for most of these objects.
- However, some applications, primarily those with custom views, have to do their own event handling.
- They have to analyze the stream of touch objects in a multitouch sequence and determine the intention of the user.

Gestures…

- Most event-handling views seek to detect common gestures that users make on their surface—things such as triple-tap, touch-and-hold (also called long press), pinching, and rotating gestures.
- The code for examining a raw stream of multitouch events and detecting one or more gestures is often complex.
- Prior to iOS 3.2, you could not reuse the code except by copying it to another project and modifying it appropriately.
- To help applications detect gestures, iOS 3.2 introduced gesture recognizers, objects that inherit directly from the UIGestureRecognizer class.

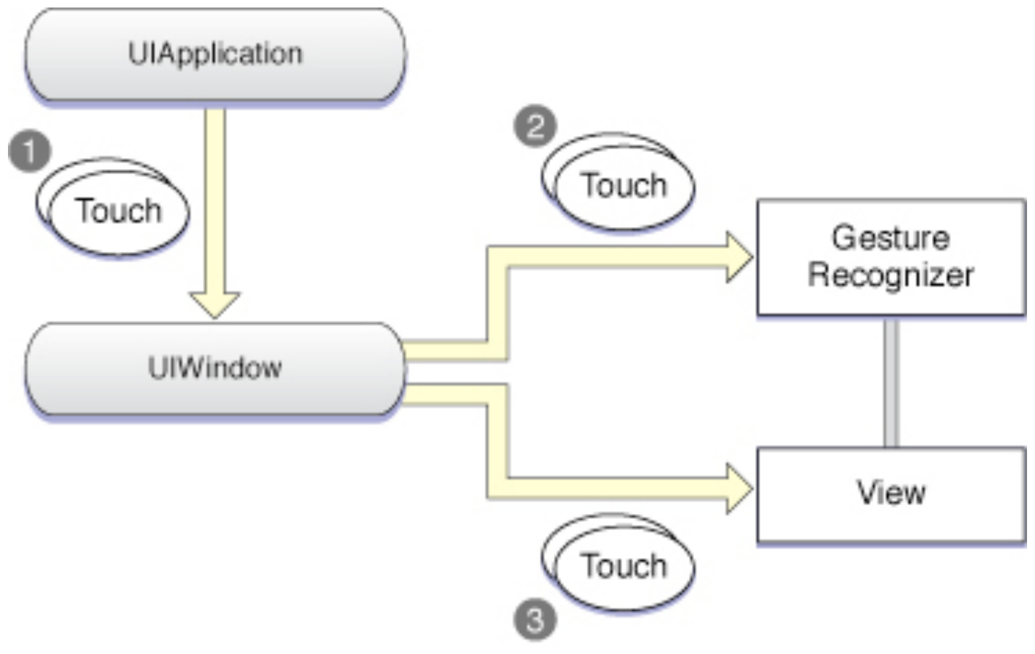
Gestures…

- Recognized gestures:
  - Tapping (any number of taps)
  - Pinching in and out (for zooming a view)
  - Panning or dragging
  - Swiping (in any direction)
  - Rotating (fingers moving in opposite directions)
  - Long press (also known as “touch and hold”)

Gestures…

- To detect its gestures, a gesture recognizer must be attached to the view that a user is touching.
- This view is known as the hit-tested view. Recall that events in iOS are represented by UIEvent objects, and each object encapsulates the UITouch objects of the current multitouch sequence.
- A set of these UITouch objects is specific to a given phase of a multitouch sequence.
- Delivery of events initially follows the usual path: from operating system to the application object to the window object representing the window in which the touches are occurring.
- But before sending an event to the hit-tested view, the window object sends it to the gesture recognizer attached to that view or to any of that view's subviews.

Gestures…



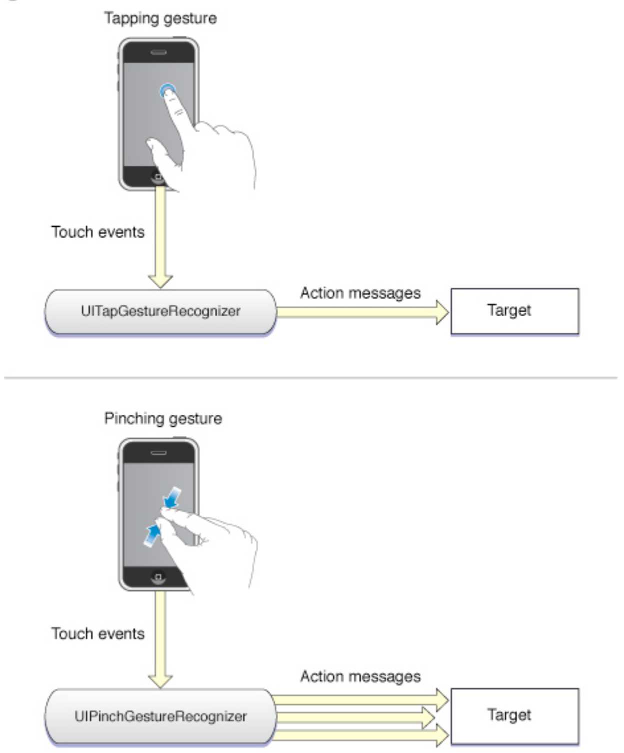
Gestures…

- Gesture recognizers act as observers of touch objects sent to their attached view or view hierarchy. However, they are not part of that view hierarchy and do not participate in the responder chain.
- Gesture recognizers may delay the delivery of touch objects to the view while they are recognizing gestures, and by default they cancel delivery of remaining touch objects to the view once they recognize their gesture.

Gestures…

- When a gesture recognizer recognizes its gesture, it sends one or more action messages to its target or multiple action messages until the gesture ends.
- This behavior is determined by whether the gesture is discrete or continuous.
- A discrete gesture, such as a double-tap, happens just once; when a gesture recognizer recognizes a discrete gesture, it sends its target a single action message.
- A continuous gesture, such as pinching, takes place over a period and ends when the user lifts the final finger in the multitouch sequence. The gesture recognizer sends action messages to its target at short intervals until the multitouch sequence ends.

Gestures…



Implementing Gesture

- To implement gesture recognition, you create a gesture-recognizer instance to which you assign a target, action, and in some cases, gesture-specific attributes.
- You attach this object to a view and then implement the action method in your target object that handles the gesture.
- To create a gesture recognizer, you must allocate and initialize an instance of a concrete UIGestureRecognizer subclass.

Target-Action Mechanism

- Target-Action Mechanism
  - A pointer to the object (target) that should receive the message
  - The selector (action) representing the action method

```
myView.addGestureRecognizer(UIPanGestureRecognizer(
    target: self,
    action: #selector(DisperseViewController.handlePan(_:)))
```

Responding to Gestures

- To handle a gesture, the target for the gesture recognizer must implement a method corresponding to the action selector specified when you initialized the gesture recognizer.
- For discrete gestures, such as a tapping gesture, the gesture recognizer invokes the method once per recognition; for continuous gestures, the gesture recognizer invokes the method at repeated intervals until the gesture ends (that is, the last finger is lifted from the gesture recognizer's view).

```
 func handlePan(recognizer: UIPanGestureRecognizer) {
     // code to react to gesture
}
```