UIGestureRecognizer and UIMenuController

BNRG CHAPTER 19

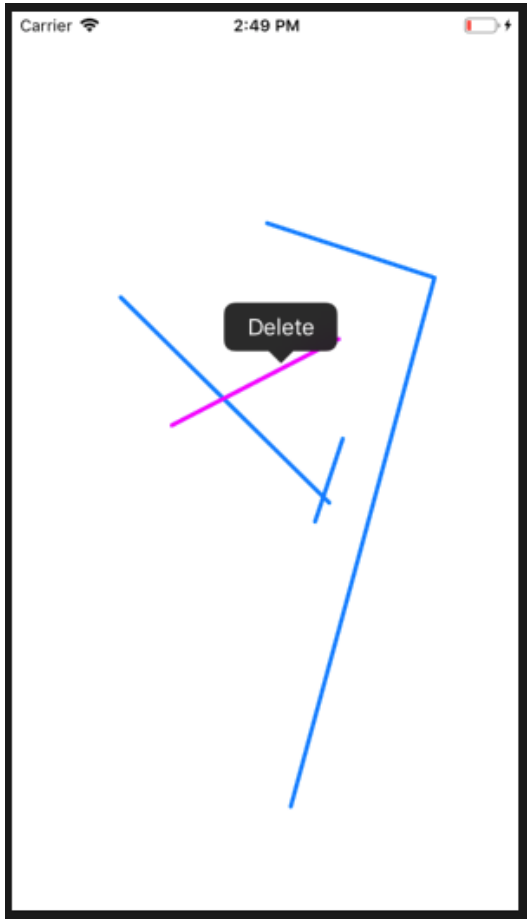
Topics

- Other gesture recognizers
- `UIMenuController`

Gesture Recognizers

`UIGestureRecognizer` gives you a chance to intercept specific kinds of gestures, such as a pinch or a swipe, and handle them in a specific way

- this means that we don't have to use the “touches” methods from the previous chapter to recognize these specific gestures



Examples of other gestures:

- double tap
- long press
- pan

`UIGestureRecognizer` Subclasses

There are specific subclasses of `UIGestureRecognizer` to recognize specific gestures

- e.g., `UITapGestureRecognizer` can recognize multi-tap gestures

The gesture recognizer intercepts touches on a view

- and if it successfully recognizes the gesture you've set up, then the touch methods don't even get sent to that view

Detecting Taps with UITapGestureRecognizer

To detect a two-tap touch event:

- create and configure a `UITapGestureRecognizer` and add it to `DrawView`

```
let doubleTapRecognizer = UITapGestureRecognizer(
            target: self,
            action: #selector(DrawView.doubleTap(_:)))
doubleTapRecognizer.numberOfTapsRequired = 2
addGestureRecognizer(doubleTapRecognizer)
```

This will call `DrawView.doubleTap(_:)` whenever a double-tap occurs on an instance of `DrawView`

`#selector(functionName())`

The underlying UI libraries are written in Objective-C

`#selector` is a way of specifying the name of a function to a the Objective-C code that implements the initializer of `UITapGestureRecognizer`

- also enables a compile-time check to see whether the specified function has actually been defined

The tap-gesture recognizer declaration creates a target-action pair

- the action is the function we specify
- the target is "self", which in this case is the `DrawView`

Update from the Book

Starting with Swift version 4, the method referenced by `#selector()` must be decorated with `@objc`

```
@objc
func doubleTap(_ gestureRecognizer: UIGestureRecognizer) {
    print("recognized a double tap")
    currentLines.removeAll()
    finsishedLines.removeAll()
    setNeedsDisplay()
}
```

Single Touch vs. Double Touch

The first touch of a double touch will have this effect:

- `touchesBegan(_:with:)` will be called on the view

But, this is not the behavior we want

- we want a single touch and a double touch to have different actions

Single Touch vs. Double Touch

Solution:

- the `UITapGestureRecognizer` should temporarily delay the call to `touchesBegan(_:with:)` until it has decided whether it will process the touch events
- and when it does decide to process the touch event, it should consume the event so that `touchesBegan(_:with:)` will not be called

```
doubleTapRecognizer.delaysTouchesBegin = true
```

Multiple Gesture Recognizers

Next bit of function: allow the user to tap on a line to select it

- this will require a new `UITapGestureRecognizer`
- configured to recognize a single tap

```
let tapRecognizer = UITapGestureRecognizer(
            target: self,
            action: #selector(DrawView.tap(_:)))
```

Multiple Gesture Recognizers

Next problem: a double tap will be recognized by both the single-tap recognizer and the double-tap recognizer

The solution will be to impose a dependency:

- the single-tap recognizer will recognize a tap only if the double-tap recognizer did not recognize the tap as part of a two-tap event

```
// tabRecognizer is the single-tap gesture recognizer
tapRecognizer.require(toFail: doubleTapRecognizer)
```

Selecting an Existing Line

Use the single-tap recognizer to enable the user to pick an existing line

- get the (x, y) location of the single tap from the `UITapGestureRecognizer`

```
let point = gestureRecognizer.location(in: self)
```

Then look at all lines

- if the tap location is close to one of the lines, then select it

Selecting an Existing Line

The book shows a nifty function: `stride(from:to:by:)`

- and uses a crude algorithm for determining whether a point is near a line
- IMHO, the geometry calculation should be a method on `Line`
- then we would ask each line "how far is the tap point from you?"
- and pick the line that's closest, provided the min value is less than some threshold

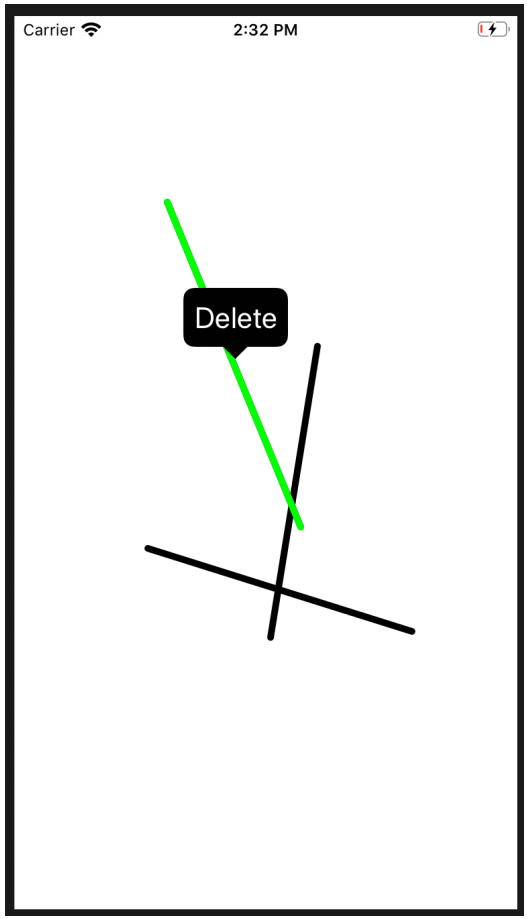
UIMenuController

Next: if the user taps near an existing line, pop up a small menu at the location of the tap

- `UIMenuController` displays and manages one of these little menus
- it is configured with a list of `UIMenuItem` objects
- each of these objects has a title and an action

A view that responds to at least one action message in the `UIMenuController`'s menu items must be the first responder of the window

- And since `DrawView` subclasses `UIView`, we must explicitly state that our view can become the first responder



Long-Press Gesture Recognizer

Next: use a long press to let the user select and drag a line

- when the long press begins, select the closest line
- then drag it

And when the long press ends, deselect the line

Long-Press Gesture Recognizer

A long press (press and hold) is a continuous gesture

- unlike a tap gesture, which is a discrete gesture

Long-Press Gesture Recognizer

To keep track of a long press, which occurs over time:

- we have to check the recognizer's state property
- when the user touches a view, the recognizer notices a possible long press: the state property will be set to `UIGestureRecognizerState.possible`
- when the user has held the touch long enough (the default is half a second), the property will be set to `UIGestureRecognizerState.began`
- when the gesture is over (the user removes his/her finger), the gesture has ended, and the state will be `UIGestureRecognizerState.ended`

At each transition, the recognizer sends its action message to its target

- and the target method can check the state of the gesture recognizer to see which transition triggered the action

UIPanGestureRecognizer

To move a selected line, we'll listen for a pan gesture

- using a `UIPanGestureRecognizer`

By default, a `UIPanGestureRecognizer` will consume any touch gesture that it recognizes

- if we don't change this, then the "tap – move – release" gesture, which draws a line, would not work
- since the `UIPanGestureRecognizer` would consume the initial tap

UIPanGestureRecognizer

Here's the code:

```
moveRecognizer = UIPanGestureRecognizer(
       target: self,
       action: #selector(DrawView.moveLine(_:)))
moveRecognizer.cancelsTouchesInView = false
addGestureRecognizer(moveRecognizer)
```

Simultaneous Recognizers

Tap – hold – move is recognized as a long press

- and not as a pan

This is because the long-press recognizer consumes the gesture

- and doesn't let any other recognizers have a chance to recognize it

Simultaneous Recognizers

Solution: we will need simultaneous recognizers

- must do this with a delegation protocol

Basically, a recognizer tells the delegate that there is a second recognizer

- and that this second recognizer should also have a chance to recognize its gesture

Simultaneous Recognizers

Here, we'll let the long-press recognizer tell the delegate that the pan recognizer should also have a chance to recognize the gesture

```
class DrawView: UIView, UIGestureRecognizerDelegate {
    // intervening code
    func gestureRecognizer(
            _ gestureRecognizer: UIGestureRecognizer,
            shouldRecognizeSimultaneouslyWith otherGestureRecognizer:
UIGestureRecognizer) -> Bool {
    return true
}
```

moveLine(_:)

Finally: during the pan gesture, move a line that's been selected

The pan gesture recognizer can tell us how far the pan has moved

Use this value to translate the begin and end coordinates of the selected line:

```
let translation = gestureRecognizer.translation(in: self)
finishedLines[index].begin.x += translation.x
finishedLines[index].begin.y += translation.y
finishedLines[index].end.x += translation.x
finishedLines[index].end.y += translation.y
gestureRecognizer.setTranslation(CGPoint.zero, in: self)
```

Other Gesture Recognizers

There are four additional gesture recognizers:

- `UIPinchGestureRecognizer`
- `UISwipeGestureRecognizer`
- `UIScreenEdgePanGestureRecognizer` ("a discrete gesture recognizer that interprets panning gestures that start near an edge of the screen.")
- `UIRotationGestureRecognizer`