Gestures

Mobile Application Development in iOS

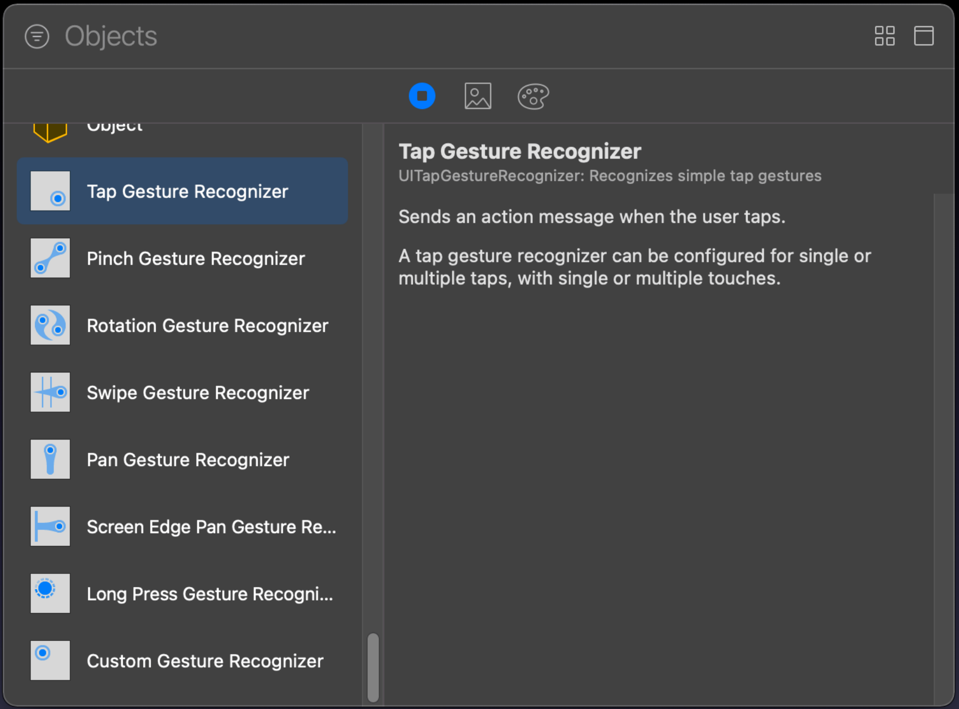
School of EECS

Washington State University

Instructor: Larry Holder

Outline

- Gestures
- Gesture recognizers
- Gesture states
- Custom gestures


Objects
Object
Tap Gesture Recognizer
Pinch Gesture Recognizer
Rotation Gesture Recognizer
Swipe Gesture Recognizer
Pan Gesture Recognizer
Screen Edge Pan Gesture Re...
Long Press Gesture Recogni...
Custom Gesture Recognizer

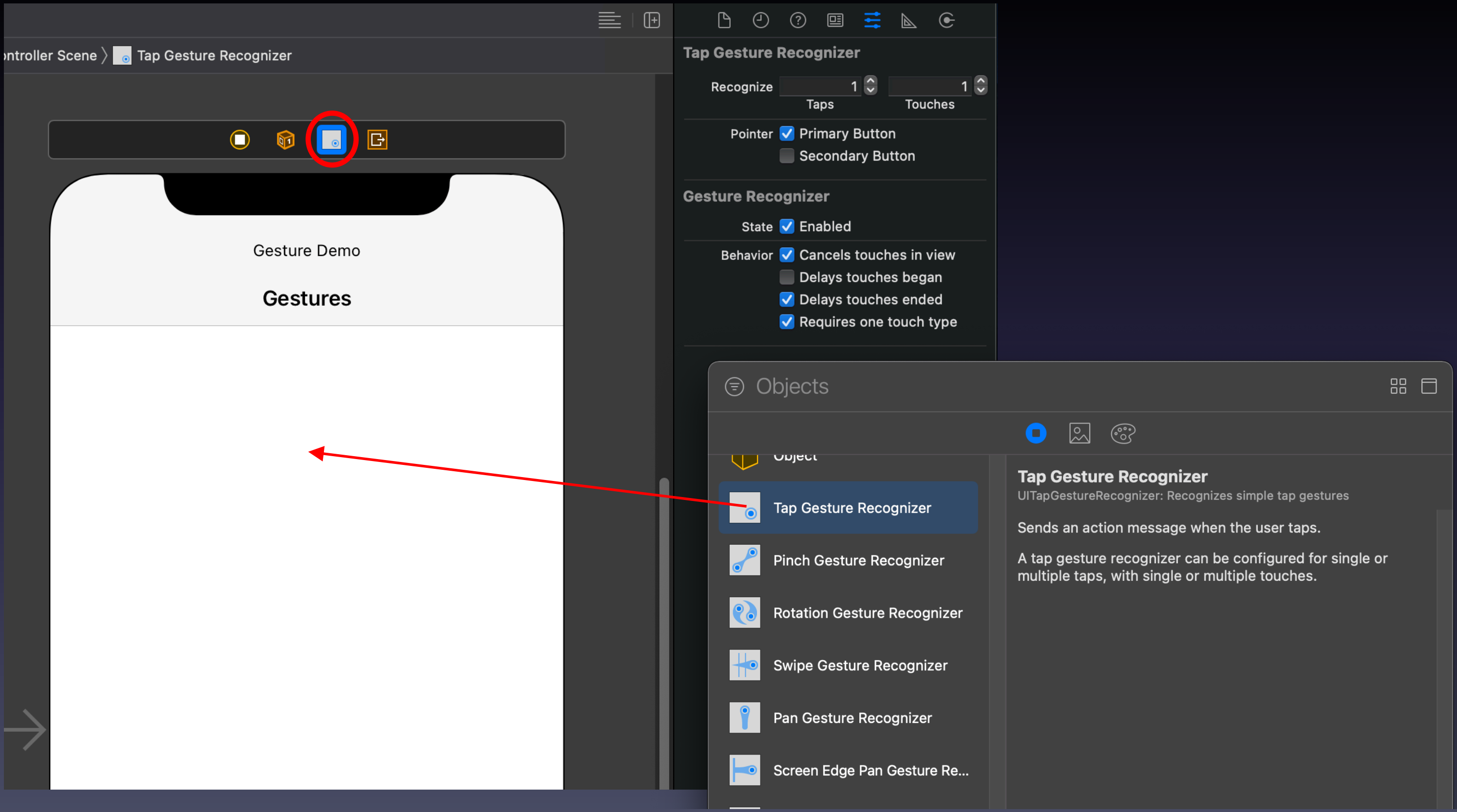
Tap Gesture Recognizer
UITapGestureRecognizer: Recognizes simple tap gestures

Sends an action message when the user taps.

A tap gesture recognizer can be configured for single or multiple taps, with single or multiple touches.

Add Gesture in Storyboard

- Step 1: Drag gesture into view



Tap Gesture Recognizer

Recognize: 1 Taps, 1 Touches

Pointer: [✓] Primary Button, [ ] Secondary Button

Gesture Recognizer

State: [✓] Enabled

Behavior: [✓] Cancels touches in view, [ ] Delays touches began, [✓] Delays touches ended, [✓] Requires one touch type

Tap Gesture Recognizer
UITapGestureRecognizer: Recognizes simple tap gestures

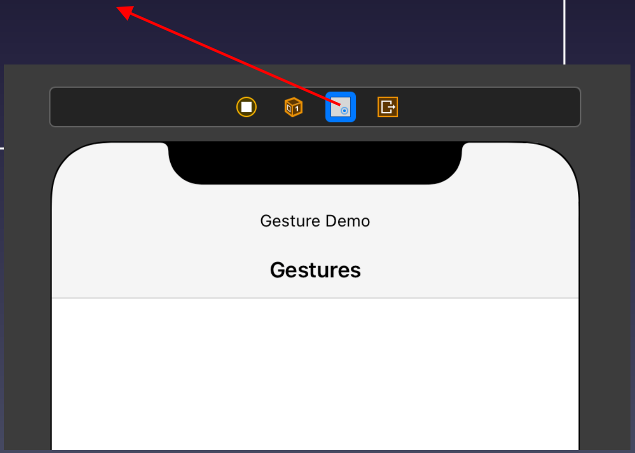
Sends an action message when the user taps.

A tap gesture recognizer can be configured for single or multiple taps, with single or multiple touches.

Add Gesture in Storyboard

- Step 2: Connect gesture to @IBAction

```
// In ViewController class...

@IBAction func tapDetected(_ sender: UIGestureRecognizer) {
    let point = sender.location(in: self.view)
    let x = Int(point.x)
    let y = Int(point.y)
    print("tap detected at (\(x),\(y))")
}
```



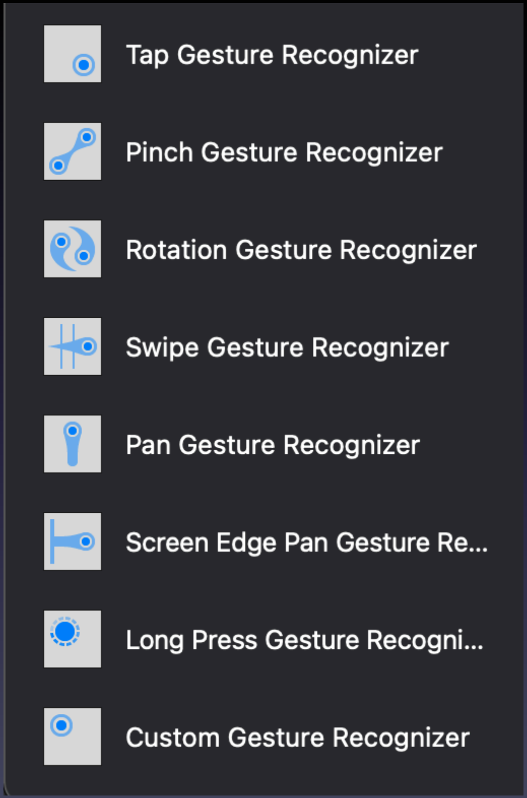
Add Gesture Programmatically

```
class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view
        let twoTouchTapGestureRecognizer =
            UITapGestureRecognizer(target: self,
                action: #selector(handleTwoTouchTap))
        twoTouchTapGestureRecognizer.numberOfTouchesRequired = 2
        self.view.addGestureRecognizer(twoTouchTapGestureRecognizer)
    }

    @objc func handleTwoTouchTap(_ sender: UITapGestureRecognizer) {
        let center = sender.location(in: self.view)
        let touch1 = sender.location(ofTouch: 0, in: self.view)
        let touch2 = sender.location(ofTouch: 1, in: self.view)
        let xc = Int(center.x), yc = Int(center.y)
        let x1 = Int(touch1.x), y1 = Int(touch1.y)
        let x2 = Int(touch2.x), y2 = Int(touch2.y)
        print("two-touch tap detected at (\(x1),\(y1)) and (\(x2),\(y2)),
            centered at (\(xc),\(yc))")
    }
}
```

Shift-Option to demo in simulator.

Other Gestures: Subclasses of UIGestureRecognizer

- UITapGestureRecognizer (multiple taps/touches)
- UIPinchGestureRecognizer
- UIRotationGestureRecognizer
- UISwipeGestureRecognizer (up, down, left, right)
- UIPanGestureRecognizer
- UIScreenEdgePanGestureRecognizer (top, bottom, left, right, all)
- UILongPressGestureRecognizer
- Custom: class MyGesture: UIGestureRecognizer


Tap Gesture Recognizer
Pinch Gesture Recognizer
Rotation Gesture Recognizer
Swipe Gesture Recognizer
Pan Gesture Recognizer
Screen Edge Pan Gesture Re...
Long Press Gesture Recogni...
Custom Gesture Recognizer

Multiple Gestures

- By default, only one gesture detected per user interaction
- Allow simultaneous gestures
  - func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer, shouldRecognizeSimultaneouslyWith otherGestureRecognizer: UIGestureRecognizer) -> Bool
- Gesture preference
  - func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer, shouldRequireFailureOf otherGestureRecognizer: UIGestureRecognizer) -> Bool

Multiple Gestures

```
class ViewController: UIViewController, UIGestureRecognizerDelegate {

     override func viewDidLoad() {
          super.viewDidLoad()
          // Do any additional setup after loading the view
          let panGestureRecognizer = UIPanGestureRecognizer(target: self,
                    action: #selector(handlePan))
          panGestureRecognizer.delegate = self
          self.view.addGestureRecognizer(panGestureRecognizer)
          let swipeGestureRecognizer = UISwipeGestureRecognizer(target: self,
                    action: #selector(handleSwipe)) // default direction = .right
          self.view.addGestureRecognizer(swipeGestureRecognizer)
     }

     func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
          shouldRecognizeSimultaneouslyWith otherGestureRecognizer: UIGestureRecognizer)
               -> Bool {
          if gestureRecognizer is UIPanGestureRecognizer { // gesture sending message
               if otherGestureRecognizer is UISwipeGestureRecognizer {
                    return true
               }
          }
          return false
     }
}
```

Multiple Gestures

```
func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
shouldRequireFailureOf otherGestureRecognizer: UIGestureRecognizer)
     -> Bool {
     if gestureRecognizer is UIPanGestureRecognizer {
          if otherGestureRecognizer is UISwipeGestureRecognizer {
               return true
          }
     }
     return false
}
```

Gesture States

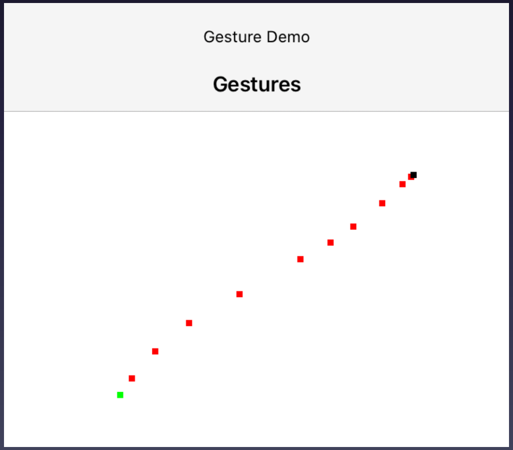
- UIGestureRecognizer.State
  - .possible (default)
  - .began
  - .changed
  - .ended (resets to .possible)
  - .cancelled (resets to .possible)
  - .failed (resets to .possible)
  - .recognized (resets to .possible)
- developer.apple.com/documentation/uikit/uigesturerecognizer/state

Gesture States (e.g., Pan)

```swift
@objc func handlePan (_ sender: UIPanGestureRecognizer) {
    let point = sender.location(in: self.view)
    let x = Int(point.x)
    let y = Int(point.y)
    switch sender.state {
    case .began: print("pan began at (\(x),\(y))")
    case .changed: print("pan changed to (\(x),\(y))")
    case .ended: print("pan ended at (\(x),\(y))")
    default: print("pan in other state at (\(x),\(y))")
    }
}
```

SideBar: Drawing Boxes to Track Gesture

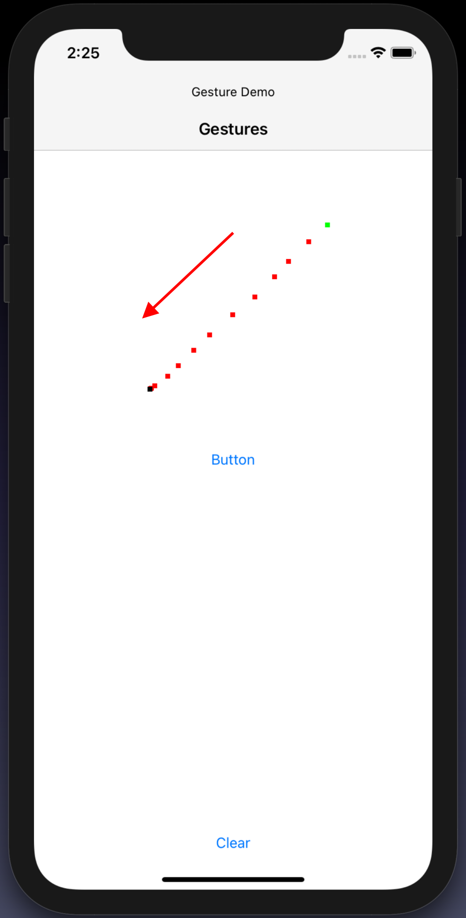
```
var boxViews: [UIView] = []

func drawBox(point: CGPoint, color: UIColor) {
    let boxRect = CGRect(x: point.x, y: point.y,
        width: 5.0, height: 5.0)
    let boxView = UIView(frame: boxRect)
    boxView.backgroundColor = color
    self.view?.addSubview(boxView)
    boxViews.append(boxView)
}

func clearBoxes() {
    for boxView in boxViews {
        boxView.removeFromSuperview()
    }
    boxViews.removeAll()
}
```

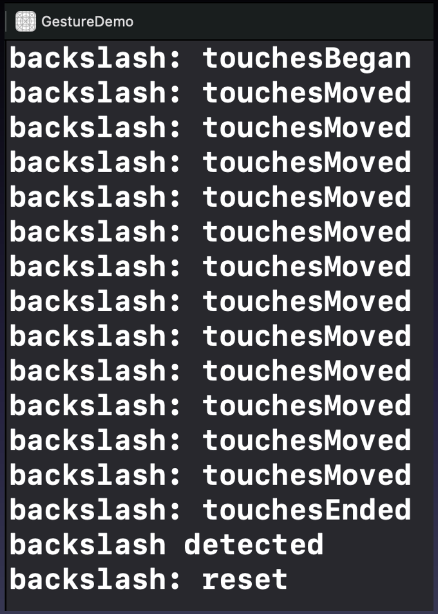


Custom Gestures

- Import UIKit and UIKit.UIGestureRecognizerSubclass
- Create subclass of UIGestureRecognizer
  - Defines methods and properties to override
- Override main gesture methods
  - touchesBegan(_ touches: Set<UITouch>, with event: UIEvent)
  - touchesMoved(_ touches: Set<UITouch>, with event: UIEvent)
  - touchesEnded(_ touches: Set<UITouch>, with event: UIEvent)
  - touchesCancelled(_ touches: Set<UITouch>, with event: UIEvent)
  - reset()

Custom Gesture Example: Backslash





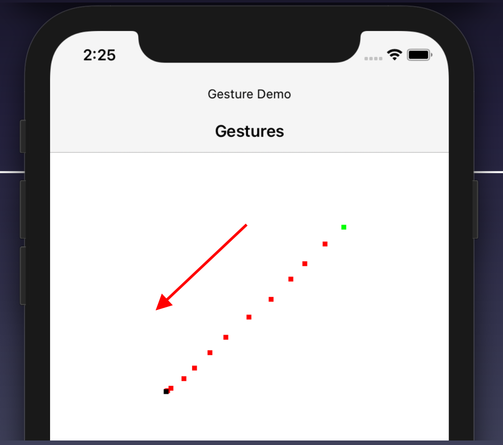
```
backslash: touchesBegan
backslash: touchesMoved
backslash: touchesMoved
backslash: touchesMoved
backslash: touchesMoved
backslash: touchesMoved
backslash: touchesMoved
backslash: touchesMoved
backslash: touchesMoved
backslash: touchesMoved
backslash: touchesMoved
backslash: touchesMoved
backslash: touchesMoved
backslash: touchesEnded
backslash detected
backslash: reset
```

Backslash Custom Gesture (1)

```
import UIKit
import UIKit.UIGestureRecognizerSubclass

class BackslashGestureRecognizer: UIGestureRecognizer {

    var minLength: Float = 100
    var initialPoint: CGPoint!
    var previousPoint: CGPoint!

    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent) {
        print("backslash: touchesBegan")
        let touch = touches.first
        if let point = touch?.location(in: self.view) {
            initialPoint = point
            previousPoint = point
            state = .began
        }
    }
```

Backslash Custom Gesture (2)

```
override func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent) {
    print("backslash: touchesMoved")
    let touch = touches.first
    if let point = touch?.location(in: self.view) {
        if ((point.x __ previousPoint.x) &&
            (point.y __ previousPoint.y)) {
            previousPoint = point
            state = .changed
        } else {
            state = .failed
        }
    }
}
```



Backslash Custom Gesture (3)

```
override func touchesEnded(_ touches: Set<UITouch>, with event: UIEvent) {
     print("backslash: touchesEnded")
     let touch = touches.first
     if let point = touch?.location(in: self.view) {
          if (point != initialPoint) &&
               (distance(point, initialPoint) >= minLength) {
               state = .ended
          } else {
               state = .failed
          }
     }
}

func distance(_ p1: CGPoint, _ p2: CGPoint) -> Float {
     let xdist = abs(p1.x - p2.x)
     let ydist = abs(p1.y - p2.y)
     let dist = sqrt((xdist * xdist) + (ydist * ydist))
     return Float(dist)
}
```

Backslash Custom Gesture (4)

```
    override func touchesCancelled(_ touches: Set<UITouch>,
                                    with event: UIEvent) {
        print("backslash: touchesCancelled")
        state = .cancelled
    }

    override func reset() {
        print("backslash: reset")
    }

}
```

Backslash Custom Gesture (5)

```swift
// In viewDidLoad...
let backslashGestureRecognizer =
    BackslashGestureRecognizer(target: self,
        action: #selector(handleBackslash))
backslashGestureRecognizer.minLength = 25
backslashGestureRecognizer.delegate = self
self.view.addGestureRecognizer(backslashGestureRecognizer)

// In ViewController...
@objc func handleBackslash(_ sender: BackslashGestureRecognizer) {
    if sender.state == .ended {
        print("backslash detected")
    }
}
```

Remember to conform your ViewController to UIGestureRecognizerDelegate

Custom Gestures

- Preserving interactions with view elements
  - E.g., button taps will go to gesture, not UIButton
  - Use gestureRecognizer: shouldReceive

```
// In ViewController
func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                             shouldReceive touch: UITouch) -> Bool {
    if touch.view is UIButton {
        return false
    }
    // Add more checks for other types of view elements, as needed
    return true
}
```

Resources

- Human Interface Guidelines: Gestures
  - developer.apple.com/design/human-interface-guidelines/ios/user-interaction/gestures
- UIGestureRecognizer API Reference
  - developer.apple.com/documentation/uikit/uigesturerecognizer
- Implementing a custom gesture recognizer
  - developer.apple.com/documentation/uikit/touches_presses_and_gestures/implementing_a_custom_gesture_recognizer