Gestures

Mobile Application Development in iOS

School of EECS

Washington State University

Instructor: Larry Holder
Outline

• Gestures
• Gesture recognizers
• Gesture states
• Custom gestures
Add Gesture in Storyboard

• Step 1: Drag gesture into view
Add Gesture in Storyboard

• Step 2: Implement `@IBAction` for gesture

```swift
// In ViewController class...

@IBAction func tapDetected (_ sender: UIGestureRecognizer) {
    if sender.state == .ended {
        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 in Storyboard

• Step 3: Connect gesture to action
Add Gesture Programmatically

• View controller class conforms to `UIGestureRecognizerDelegate`

• Implement method to handle gesture

• Create gesture recognizer and add to view
class ViewController: UIViewController, UIGestureRecognizerDelegate {

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

@objc func handleTwoTouchTap (_ sender: UITapGestureRecognizer) {
    if sender.state == .ended {
        let point = sender.location(in: self.view)
        let x = Int(point.x)
        let y = Int(point.y)
        print("two-touch tap detected at (\(x),\(y))")
    }
}
}
Other Gestures: Subclasses of `UIGestureRecognizer`

- `UITapGestureRecognizer` (multiple taps/touches)
- `UIPinchGestureRecognizer`
- `UIRotationGestureRecognizer`
- `UISwipeGestureRecognizer`
- `UIPanGestureRecognizer`
- `UIScreenEdgeGestureRecognizer`
- `UILongPressGestureRecognizer`
- class `MyGesture: UIGestureRecognizer` (custom)

Note: Only one gesture detected per user interaction.
Gesture States

- UIGestureRecognizerState
  - .possible
  - .began
  - .changed
  - .ended
  - .cancelled
  - .failed
  - .recognized
let panGestureRecognizer = UIPanGestureRecognizer(target: self, action: #selector(handlePan))
panGestureRecognizer.delegate = self
self.view.addGestureRecognizer(panGestureRecognizer)

@objc func handlePan (_ sender: UIPanGestureRecognizer) {
    let point = sender.location(in: self.view)
    let x = Int(point.x)
    let y = Int(point.y)
    if (sender.state == .began) {
        print("pan began at ((x),(y))")
    }
    if (sender.state == .changed) {
        print("pan moved to ((x),(y))")
    }
    if (sender.state == .ended) {
        print("pan ended at ((x),(y))")
    }
}
Custom Gestures

• Create subclass of UIGestureRecognizer
• Import UIKit.UIGestureRecognizerSubclass
  – Defines methods and properties to override
• Override main gesture functions
  – 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
import UIKit
import UIKit.UIGestureRecognizerSubclass

class BackSlashGesture: UIGestureRecognizer
{
    var initialPoint: CGPoint!
    var previousPoint: CGPoint!

    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent) {
        print("\n\nbackslash: touchesBegan\n\n")
        super.touchesBegan(touches, with: event)
        let touch = touches.first
        let point = (touch?.location(in: self.view))!
        initialPoint = point
        previousPoint = point
    }
}
override func touchesMoved(_ touches: Set<UITouch>,
    with event: UIEvent) {
    print("backslash: touchesMoved")
    super.touchesMoved(touches, with: event)
    let touch = touches.first
    let point = (touch?.location(in: self.view))!
    if ((point.x >= previousPoint.x) &&
        (point.y <= previousPoint.y)) {
        previousPoint = point
    } else {
        self.state = .failed
    }
}
override func touchesEnded(_ touches: Set<UITouch>,
    with event: UIEvent) {
    print("backslash: touchesEnded")
    super.touchesEnded(touches, with: event)
    let touch = touches.first
    let point = (touch?.location(in: self.view))!
    if ((self.state == .possible) && (point != initialPoint)) {
        self.state = .recognized
    } else {
        self.state = .failed
    }
}
override func touchesCancelled(_ touches: Set<UITouch>,
with event: UIEvent) {
    super.touchesCancelled(touches, with: event)
    self.state = .cancelled
}

override func reset() {
    super.reset()
}
// In viewDidLoad...
let backSlashGesture = BackSlashGesture(target: self,
    action: #selector(handleBackSlash))
backSlashGesture.delegate = self
self.view.addGestureRecognizer(backSlashGesture)

// In ViewController...
@objc func handleBackSlash(_ sender: BackSlashGesture) {
    print("backslash detected")
}

// Remember to conform ViewController to UIGestureRecognizerDelegate

Remember to conform ViewController to UIGestureRecognizerDelegate
Resources

• Human Interface Guidelines: 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_customGesture_recognizer