if let touch = touches.first {

let touchLocation = touch.location(in: self.view)

let previousTouchLocation = touch.previousLocation(in: self.view)

// ...

}

view.isMultipleTouchEnabled = true // placed in viewDidLoad

// ...

override func touchesBegan(\_ touches: Set<UITouch>, with event: UIEvent?) {

if isMultipleTouchEnabled {

for touch in touches {

let touchLocation = touch.location(in: self.view)

// Handle each touch

}

}

}

let tap = UITapGestureRecognizer(target: self,

action: #selector(handleTap))

view.addGestureRecognizer(tap)

@objc func handleTap(sender: UITapGestureRecognizer) {

print("View was tapped!")

}

let pinchRecognizer = UIPinchGestureRecognizer(target: self,

action: #selector(handlePinch))

view.addGestureRecognizer(pinchRecognizer)

@objc func handlePinch(recognizer: UIPinchGestureRecognizer) {

if recognizer.state == .changed {

let scale = recognizer.scale

recognizer.view?.transform = (recognizer.view?.transform.scaledBy(x: scale,

y: scale))!

recognizer.scale = 1.0

}

}

func gestureRecognizer (\_ gestureRecognizer: UIGestureRecognizer,

shouldRecognizeSimultaneouslyWith

otherGestureRecognizer: UIGestureRecognizer) -> Bool {

return true

}

let panGesture = UIPanGestureRecognizer(target: self,

action: #selector(handlePan))

view.addGestureRecognizer(panGesture)

@objc func handlePan(sender: UIPanGestureRecognizer) {

if let view = sender.view {

let translation = sender.translation(in: view)

view.center = CGPoint(x: view.center.x + translation.x,

y: view.center.y + translation.y)

sender.setTranslation(CGPoint.zero, in: view)

}

}

textField.keyboardType = .emailAddress // Email keyboard

textField.returnKeyType = .done // "Done" key

textField.isSecureTextEntry = **true**

NotificationCenter.default.addObserver(self, selector: #selector(keyboardWillShow),  
 name: UIResponder.keyboardWillShowNotification, object: nil)

NotificationCenter.default.addObserver(self, selector: #selector(keyboardWillHide),  
 name: UIResponder.keyboardWillHideNotification, object: nil)

@objc func keyboardWillShow(notification: NSNotification) {

if let keyboardSize = (notification.userInfo?[UIResponder.keyboardFrameEndUserInfoKey] as? NSValue)?.cgRectValue {

if view.frame.origin.y == 0 {

view.frame.origin.y -= keyboardSize.height

}

}

}

@objc func keyboardWillHide(notification: NSNotification) {

if view.frame.origin.y != 0 {

view.frame.origin.y = 0

}

}

func textFieldShouldReturn(\_ textField: UITextField) -> Bool {

textField.resignFirstResponder()

return true

}

func textField(\_ textField: UITextField, shouldChangeCharactersIn range: NSRange, replacementString string: String) -> Bool {

// Validate input or restrict characters

return true

}

func textField(\_ textField: UITextField, shouldChangeCharactersIn range: NSRange, replacementString string: String) -> Bool {

// Add your validation logic here

return true

}

class CustomTextField: UITextField {

override var inputAccessoryView: UIView? {

get {

let toolbar = UIToolbar()

toolbar.sizeToFit()

let flexSpace = UIBarButtonItem(barButtonSystemItem: .flexibleSpace,

target: nil, action: nil)

let doneButton = UIBarButtonItem(title: "Done",   
 style: .done,

target: self,

action: #selector(dismissKeyboard))

toolbar.setItems([flexSpace, doneButton], animated: true)

return toolbar

}

}

@objc func dismissKeyboard() {

resignFirstResponder()

}

}

override func becomeFirstResponder() -> Bool {

return true

}

override func motionEnded(\_ motion: UIEvent.EventSubtype, with event: UIEvent?) {

if motion == .motionShake {

// Handle shake event

}

}

let motionManager = CMMotionManager()

func startMotionUpdates() {

if motionManager.isAccelerometerAvailable {

motionManager.accelerometerUpdateInterval = 0.1

motionManager.startAccelerometerUpdates(to: .main)

{ [weak self] (data, error) in

guard let motionData = data else {

print("No data available")

return

}

// Use accelerometer data

}

}

}

if let acceleration = data?.acceleration {

let x = acceleration.x

let y = acceleration.y

let z = acceleration.z

// Process acceleration data

}

if motionManager.isGyroAvailable {

motionManager.gyroUpdateInterval = 0.1

motionManager.startGyroUpdates(to: .main) { (data, error) in

guard let motionData = data else {

print("No data available")

return

}

// Use gyroscope data

}

}

if motionManager.isDeviceMotionAvailable {

motionManager.deviceMotionUpdateInterval = 0.1

motionManager.startDeviceMotionUpdates(to: .main) { (data, error) in

guard let motionData = data else {

print("No data available")

return

}

// Use device motion data

}

}