SCIENCES SORBONNE UNIVERSITÉ

«ScreenContact»

Fabrice.Kordon@lip6.fr

LIP6

Golas of the example

Handle touches

- Retrieve them
  - **Button to activate / deactivate multitouch**
- Count them
- Display their position in an «active area»
- Check number of taps to change the color of the active area
  - **Transparent white**
  - **Transparent red**
  - **Transparent yellow**
- Display a «pressure indicator» (on some iPhones)

Remarks

- No multitouch on the simulator
- Pressure available on some iPhones only
  - **Checked unavailability: iPod touch, iPhone SE, iPad pro 9.7**

Demo

ViewController

Sake of simplicity…

Code located in a ViewController

ViewController


```
import UIKit

class ViewController: UIViewController {

    private let contacts = UILabel()
    private let positions = UITextView()
    private let sensibleArea = UIView()
    private let b = UIButton(type: .system)
    private let forcePress = UIImageView(image: UIImage(named: "pressure"))

    // to hide the statusBar
override var prefersStatusBarHidden : Bool {
    return true
}
```

ViewController


```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.
    self.view = UIView()
    self.view.backgroundColor = UIColor.white
    let background = UIImageView(image: UIImage(named: "carpet"))
    background.frame = UIScreen.main.bounds
    background.alpha = 0.7
    self.view.addSubview(background)
    contacts.text = "No contact"
    contacts.backgroundColor = .white
    contacts.textAlignment = .center
    self.view.addSubview(contacts)
    b.setTitle("Multi", for: .normal)
    b.addTarget(self, action: #selector(multiMono), for: .touchDown)
    b.backgroundColor = .white
    self.view.addSubview(b)
    positions.text = "---"
    positions.textAlignment = .center
    positions.backgroundColor = .white
    positions.isSelectable = false
    self.view.addSubview(positions)
    self.view.addSubview(sensibleArea)
    forcePress.isHidden = true;
    self.view.addSubview(forcePress)

    // no orientation managed
    let size = UIScreen.main.bounds.size
    forcePress.center = CGPoint(x: size.width / 2, y: size.height / 2)
    contacts.frame = CGRect(x: 10.0, y: 30.0,
                            width: size.width - 80.0, height: 20.0)
    b.frame = CGRect(x: size.width - 70.0, y: 30.0,
                     width: 60.0, height: 20.0)
    positions.frame = CGRect(x: 10.0, y: 60.0,
                             width: size.width - 20.0, height: 50.0)
    sensibleArea.frame = CGRect(x: 0.0, y: 120.0,
                                width: size.width, height: size.height - 120.0)
}
```

ViewController


```
@objc func multiMono() {
    if b.title(for: .normal) == "Multi" {
        b.setTitle("Mono", for: .normal)
        sensibleArea.isMultipleTouchEnabled = true
    } else {
        b.setTitle("Multi", for: .normal)
        sensibleArea.isMultipleTouchEnabled = false
    }
}
```

ViewController


```
// Service function
func handlingTouches(touches: Set<UITouch>, event: UIEvent?) {
    let myTouches = event?.touches(for: sensibleArea)
    let nbTouches = myTouches?.count
    var myPositions = ""
    if nbTouches != nil && nbTouches! > 0  {
        sensibleArea.backgroundColor =
            UIColor(red: 1.0, green: 1.0, blue: 1.0, alpha: 0.5)
        if nbTouches! == 1 {
            contacts.text = "One touch"
        } else {
            contacts.text = String(format: "%d touches", nbTouches!)
        }
        for aTouch in myTouches! {
            let pos = aTouch.location(in: sensibleArea)
            myPositions = myPositions + " (\(pos.x),\(pos.y))"
        }
        positions.text = myPositions
    }
    if nbTouches != nil && myTouches!.first?.tapCount == 2 {
        sensibleArea.backgroundColor =
            UIColor(red: 1.0, green: 0.0, blue: 0.0, alpha: 0.5)
    }
    if nbTouches != nil && myTouches!.first?.tapCount == 3 {
        sensibleArea.backgroundColor =
            UIColor(red: 1.0, green: 1.0, blue: 0.0, alpha: 0.5)
    }
    if nbTouches != nil && myTouches!.first!.force > 2.0 {
        forcePress.isHidden = false
    } else {
        forcePress.isHidden = true
    }
}
```

ViewController


```
// UIResponder

override func touchesBegan(_ touches: Set<UITouch>,
                           with event: UIEvent?) {
    self.handlingTouches(touches: touches, event: event)
}

override func touchesEnded(_ touches: Set<UITouch>,
                           with event: UIEvent?) {
    contacts.text = "No contact"
    positions.text = "---"
    sensibleArea.backgroundColor = UIColor.clear
    forcePress.isHidden = true
}

override func touchesMoved(_ touches: Set<UITouch>,
                           with event: UIEvent?) {
    self.handlingTouches(touches: touches, event: event)
}

override func touchesCancelled(_ touches: Set<UITouch>,
                               with event: UIEvent?) {
    sensibleArea.backgroundColor = UIColor.clear
    forcePress.isHidden = true
}
}
```

As a conclusion…

You know now how to handle touch/multitouch

- Offers numerous possibilities
  - **Like in games**

Important remark (multitouch)

- You may have noticed it already from the demos…
  - **No order is guaranteed in the touches (this is a set)**
  - **The order may change any time too**
  - **Be aware if you want to track one touch**