SCIENCES SORBONNE UNIVERSITÉ

# «iDraw»

Fabrice.Kordon@lip6.fr

LIP6

# Goal of the example



## A very basic drawing App

- Device with no pencil and no force touch
  - **Double tap for draw**
  - **Single tap to erase**
- Device with no pencil and force touch
  - **Double tap for draw**
  - **Single tap to erase**
  - **Handling of the pressure to enlarge the drawing**
- Device with pencil
  - **Pencil to draw**
  - **Finger to erase**
  - **Support of the pencil orientation and force**

# Demo (iPod touch)

# Demo (iPhone)

# Demo (iPad Pro)

# ViewController



**Sake of simplicity…**

Code located in a ViewController

# ViewController



```
import UIKit

class ViewController: UIViewController {
    private let currentState = UIImageView()
    private let sheet = UIImageView(frame: UIScreen.main.bounds)

    private let eraserColor = UIColor.white
    private var pencilColor = UIColor.black

    private let crtColor = UISegmentedControl(items: ["black", "blue", "red", "orange"])
private let piBy2 = CGFloat.pi / 2

// No need for the status bar
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(frame: UIScreen.main.bounds)
    self.view.backgroundColor = UIColor.white
    self.view.isMultipleTouchEnabled = false
    self.view.addSubview(sheet)
    self.view.addSubview(currentState)
    self.view.addSubview(crtColor)
    crtColor.addTarget(self, action: #selector(updateColor(sender:)),
                       for: .valueChanged)
    crtColor.selectedSegmentIndex = 0
    self.displayInSize(size: UIScreen.main.bounds.size)
}
```

# ViewController



```
override func viewWillTransition(to size: CGSize,
                  with coordinator: UIViewControllerTransitionCoordinator) {
    self.displayInSize(size: size)
}

func displayInSize (size: CGSize) {
    var top = 15;
    var left = 15
    if UIDevice.current.userInterfaceIdiom == .phone &&
        size.height >= 812 {
        top = 45
    } else if UIDevice.current.userInterfaceIdiom == .phone &&
        size.width > size.height {
        top = 10
        if size.width >= 812 {
            left = 45
        }
    }
    currentState.frame = CGRect(x: left, y: top, width: 50, height: 50)
    crtColor.frame = CGRect(x: Int(size.width - crtColor.frame.size.width) - left,
                            y: top,
                            width: Int(crtColor.frame.size.width),
                            height: 30)
    sheet.frame = CGRect(x: 0, y: 0, width: size.width, height: size.height)
}
```

# ViewController



```
@objc func updateColor(sender: UISegmentedControl) {
    switch sender.selectedSegmentIndex {
    case 0: pencilColor = UIColor.black
    case 1: pencilColor = UIColor.blue
    case 2: pencilColor = UIColor.red
    case 3: pencilColor = UIColor.orange
    default: () // unreachable
    }
}
```

# ViewController



```
private func drawIt (ctx: CGContext?, touch: UITouch) {
    let prevPos = touch.previousLocation(in: self.view)
    let crtPos = touch.location(in: self.view)
    var lineWidth : CGFloat
    if touch.type == .stylus {
        lineWidth = 1 + 8.0 * (piBy2 - touch.altitudeAngle)
        pencilColor.setStroke()
        let forceMin : CGFloat = 0.0
        let forceMax : CGFloat = 3.0
        ctx?.setAlpha((touch.force - forceMin) / (forceMax - forceMin))
    } else if touch.tapCount == 2 { // degraded mode for force touch
        lineWidth = 1 + 8.0 * touch.force
        pencilColor.setStroke()
    } else {
        lineWidth = touch.majorRadius // typical size of a contact (from hardware)
        eraserColor.setStroke()
    }
    ctx?.setLineWidth(lineWidth)
    ctx?.setLineCap(.round)
    // draw between two points
    ctx?.move(to: CGPoint(x: prevPos.x, y: prevPos.y))
    ctx?.addLine(to: CGPoint(x: crtPos.x, y: crtPos.y))
    ctx?.strokePath()
  }
}
```

# ViewController



```
override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
    if touches.first?.type == .stylus ||
       touches.first?.tapCount == 2 {
        currentState.image = UIImage(named: "pencil")
    } else if touches.first?.type == .direct {
        currentState.image = UIImage(named: "eraser")
    }
}

override func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent?) {
    // Setup dthe drawinf zone + fetch the context
    UIGraphicsBeginImageContextWithOptions(UIScreen.main.bounds.size,false, 0.0)
    let contexte = UIGraphicsGetCurrentContext()
    sheet.image?.draw(in: UIScreen.main.bounds)
    drawIt(ctx: contexte, touch: touches.first!)
    // update the image + close the context
    sheet.image = UIGraphicsGetImageFromCurrentImageContext()
    UIGraphicsEndImageContext()
}

override func touchesCancelled(_ touches: Set<UITouch>, with event: UIEvent?) {
    currentState.image = nil
}

override func touchesEnded(_ touches: Set<UITouch>, with event: UIEvent?) {
    currentState.image = nil
}
```

# As a conclusion…



## This is it!

- You know now how to handle touches
  - **Fingers**
  - **Pencil**
  - **Pressure (from finger or pencil)**
- You may have fun with it!