Lecture 13 Demo Code: FaceIt Animation

Objective

Included below is the source code for the demo in lecture. It is provided under the same Creative Commons licensing as the rest of CS193p's course materials. Some of the code from previous lectures is included with unchanged portions grayed out. See Lectures 4 through 6 FaceIt Demo Code documents for the rest of the FaceIt code. And here is the complete project.

```swift
//
//  BlinkingFaceViewController.swift
//  FaceIt
//
//  Created by CS193p Instructor.
//  Copyright © 2017 Stanford University. All rights reserved.
//

import UIKit

class BlinkingFaceViewController: FaceViewController
{
    var blinking = false {
        didSet {
            blinkIfNeeded()
        }
    }

    private struct BlinkRate {
        static let closedDuration: TimeInterval = 0.4
        static let openDuration: TimeInterval = 2.5
    }


    private func blinkIfNeeded() {
        if blinking {
            faceView.eyesOpen = false
            Timer.scheduledTimer(withTimeInterval: BlinkRate.closedDuration, repeats: false) { [weak self] timer in
                self?.faceView.eyesOpen = true
                Timer.scheduledTimer(withTimeInterval: BlinkRate.openDuration, repeats: false) { [weak self] timer in
                    self?.blinkIfNeeded()
                }
            }
        }
    }

    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
        blinking = true
    }

    override func viewWillDisappear(_ animated: Bool) {
        super.viewWillDisappear(animated)
        blinking = false
    }
}
```

```swift
//
//  ViewController.swift
//  FaceIt
//
//  Created by CS193p Instructor.
//  Copyright © 2017 Stanford University. All rights reserved.
//

import UIKit

class FaceViewController: VCLLoggingViewController
{
    var expression = FacialExpression(eyes: .open, mouth: .neutral) {
        didSet {
            updateUI()
        }
    }

    @IBOutlet weak var faceView: FaceView! {
        didSet {
            let handler = #selector(FaceView.changeScale(byReactingTo:))
            let pinchRecognizer = UIPinchGestureRecognizer(target: faceView, action: handler)
            faceView.addGestureRecognizer(pinchRecognizer)
//            let tapRecognizer = UITapGestureRecognizer(target: self, action: #selector(toggleEyes(byReactingTo:)))
//            tapRecognizer.numberOfTapsRequired = 1
//            faceView.addGestureRecognizer(tapRecognizer)
            let swipeUpRecognizer = UISwipeGestureRecognizer(target: self, action: #selector(increaseHappiness))
            swipeUpRecognizer.direction = .up
            faceView.addGestureRecognizer(swipeUpRecognizer)
            let swipeDownRecognizer = UISwipeGestureRecognizer(target: self, action: #selector(decreaseHappiness))
            swipeDownRecognizer.direction = .down
            faceView.addGestureRecognizer(swipeDownRecognizer)
            updateUI()
        }
    }

    func increaseHappiness()
    {
        expression = expression.happier
    }
    func decreaseHappiness()
    {
        expression = expression.sadder
    }

    func toggleEyes(byReactingTo tapRecognizer: UITapGestureRecognizer) {
        if tapRecognizer.state == .ended {
            let eyes: FacialExpression.Eyes = (expression.eyes == .closed) ? .open : .closed
            expression = FacialExpression(eyes: eyes, mouth: expression.mouth)
        }
    }

    private func updateUI()
    {
        switch expression.eyes {
        case .open:
            faceView?.eyesOpen = true
        case .closed:
            faceView?.eyesOpen = false
        case .squinting:
            faceView?.eyesOpen = false
        }
        faceView?.mouthCurvature = mouthCurvatures[expression.mouth] ?? 0.0
    }

    private let mouthCurvatures =
        [FacialExpression.Mouth.grin:0.5,.frown:-1.0,.smile:1.0,.neutral:0.0,.smirk:-0.5]
    // MARK: Head Shake

    private struct HeadShake {
        static let angle = CGFloat.pi/6                  // radians
        static let segmentDuration: TimeInterval = 0.5  // each head shake has 3 segments
    }

    private func rotateFace(by angle: CGFloat) {
        faceView.transform = faceView.transform.rotated(by: angle)
    }

    private func shakeHead() {
        UIView.animate(
            withDuration: HeadShake.segmentDuration,
            animations: { self.rotateFace(by: HeadShake.angle) },
            completion: { finished in
                if finished {
                    UIView.animate(
                        withDuration: HeadShake.segmentDuration,
                        animations: { self.rotateFace(by: -HeadShake.angle*2) },
                        completion: { finished in
                            if finished {
                                UIView.animate(
                                    withDuration: HeadShake.segmentDuration,
                                    animations: { self.rotateFace(by: HeadShake.angle) }
                                )
                            }
                        }
                    )
                }
            }
        )
    }

    @IBAction func shakeHead(_ sender: UITapGestureRecognizer) {
        shakeHead()
    }
}
```