Lecture 4 Demo Code: FaceIt

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. And here is the complete project.

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

import UIKit

@IBDesignable
class FaceView: UIView
{
    // Public API

    @IBInspectable
    var mouthCurvature: Double = 0.5 // 1.0 is full smile and -1.0 is full frown

    @IBInspectable
    var eyesOpen: Bool = true

    @IBInspectable
    var scale: CGFloat = 0.9

    @IBInspectable
    var lineWidth: CGFloat = 5.0

    @IBInspectable
    var color: UIColor = UIColor.blue

    // Private Implementation

    private struct Ratios {
        static let skullRadiusToEyeOffset: CGFloat = 3
        static let skullRadiusToEyeRadius: CGFloat = 10
        static let skullRadiusToMouthWidth: CGFloat = 1
        static let skullRadiusToMouthHeight: CGFloat = 3
        static let skullRadiusToMouthOffset: CGFloat = 3
    }

    private var skullRadius: CGFloat {
        return min(bounds.size.width, bounds.size.height) / 2 * scale
    }

    private var skullCenter: CGPoint {
        return CGPoint(x: bounds.midX, y: bounds.midY)
    }
```

```
private enum Eye {
    case left
    case right
}

private func pathForEye(_ eye: Eye) -> UIBezierPath
{
    func centerOfEye(_ eye: Eye) -> CGPoint {
        let eyeOffset = skullRadius / Ratios.skullRadiusToEyeOffset
        var eyeCenter = skullCenter
        eyeCenter.y -= eyeOffset
        eyeCenter.x += ((eye == .left) ? -1 : 1) * eyeOffset
        return eyeCenter
    }

    let eyeRadius = skullRadius / Ratios.skullRadiusToEyeRadius
    let eyeCenter = centerOfEye(eye)

    let path: UIBezierPath
    if eyesOpen {
        path = UIBezierPath(
            arcCenter: eyeCenter,
            radius: eyeRadius,
            startAngle: 0,
            endAngle: CGFloat.pi * 2,
            clockwise: true
        )
    } else {
        path = UIBezierPath()
        path.move(to: CGPoint(x: eyeCenter.x - eyeRadius, y: eyeCenter.y))
        path.addLine(to: CGPoint(x: eyeCenter.x + eyeRadius, y: eyeCenter.y))
    }
    path.lineWidth = lineWidth

    return path
}


private func pathForMouth() -> UIBezierPath
{
    let mouthWidth = skullRadius / Ratios.skullRadiusToMouthWidth
    let mouthHeight = skullRadius / Ratios.skullRadiusToMouthHeight
    let mouthOffset = skullRadius / Ratios.skullRadiusToMouthOffset

    let mouthRect = CGRect(
        x: skullCenter.x - mouthWidth / 2,
        y: skullCenter.y + mouthOffset,
        width: mouthWidth,
        height: mouthHeight
    )

    let smileOffset = CGFloat(max(-1, min(mouthCurvature, 1))) * mouthRect.height

    let start = CGPoint(x: mouthRect.minX, y: mouthRect.midY)
    let end = CGPoint(x: mouthRect.maxX, y: mouthRect.midY)
    let cp1 = CGPoint(x: start.x + mouthRect.width / 3, y: start.y + smileOffset)
    let cp2 = CGPoint(x: end.x - mouthRect.width / 3, y: start.y + smileOffset)

    let path = UIBezierPath()
    path.move(to: start)
    path.addCurve(to: end, controlPoint1: cp1, controlPoint2: cp2)
    path.lineWidth = lineWidth
    return path
}
```

```
    private func pathForSkull() -> UIBezierPath {
        let path = UIBezierPath(
            arcCenter: skullCenter,
            radius: skullRadius,
            startAngle: 0,
            endAngle: 2 * CGFloat.pi,
            clockwise: false
        )
        path.lineWidth = lineWidth
        return path
    }

    override func draw(_ rect: CGRect) {
        color.set()
        pathForSkull().stroke()
        pathForEye(.left).stroke()
        pathForEye(.right).stroke()
        pathForMouth().stroke()
    }
}
```