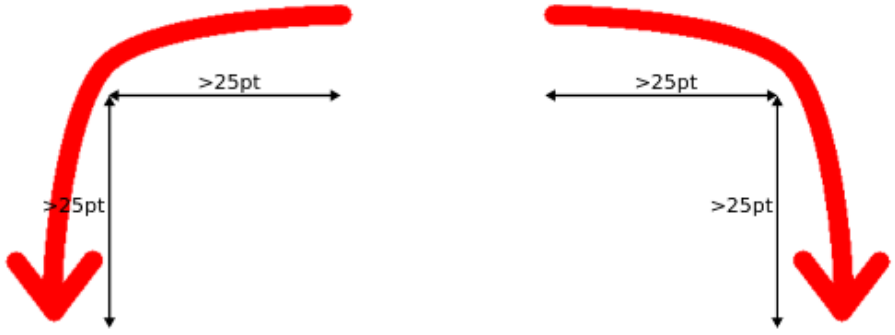
Aufgabe 11: Custom Gesture Recognizer

Andreas Hessenberger

Aufgabe:

Ios-App entwickeln, die die Rotation eines Images aufgrund von folgender Geste unterstützt:



(a) Rotation um 90° nach links. (b) Rotation um 90° nach rechts.

Abbildung 2: Darstellung der Gesten, die zur Rotation des Images führen sollen.

Folgende bedingte Abfolge einer Touch-Sequenz wird als Geste zur Einleitung der Rechtsrotation erkannt:

- Genau ein Finger trift auf das Display.
- Der Finger bewegt sich kontinuierlich mindestens 25 Punkte nach rechts. Relativ zum Punkt der ersten Berührung darf sich der Finger hier auch nach unten jedoch nicht nach oben bewegen.
- Nach Erfüllung der vorhergehenden Bedingung bewegt sich der Finger mindestens 25 Punkte nach unten. Der Finger darf sich dabei weiterhin nach rechts, jedoch nicht nach links bewegen.

Die bedingte Abfolge einer Touch-Sequenz zur Erkennung der Geste, die eine Linksrotation einleitet, erfolgt analog.

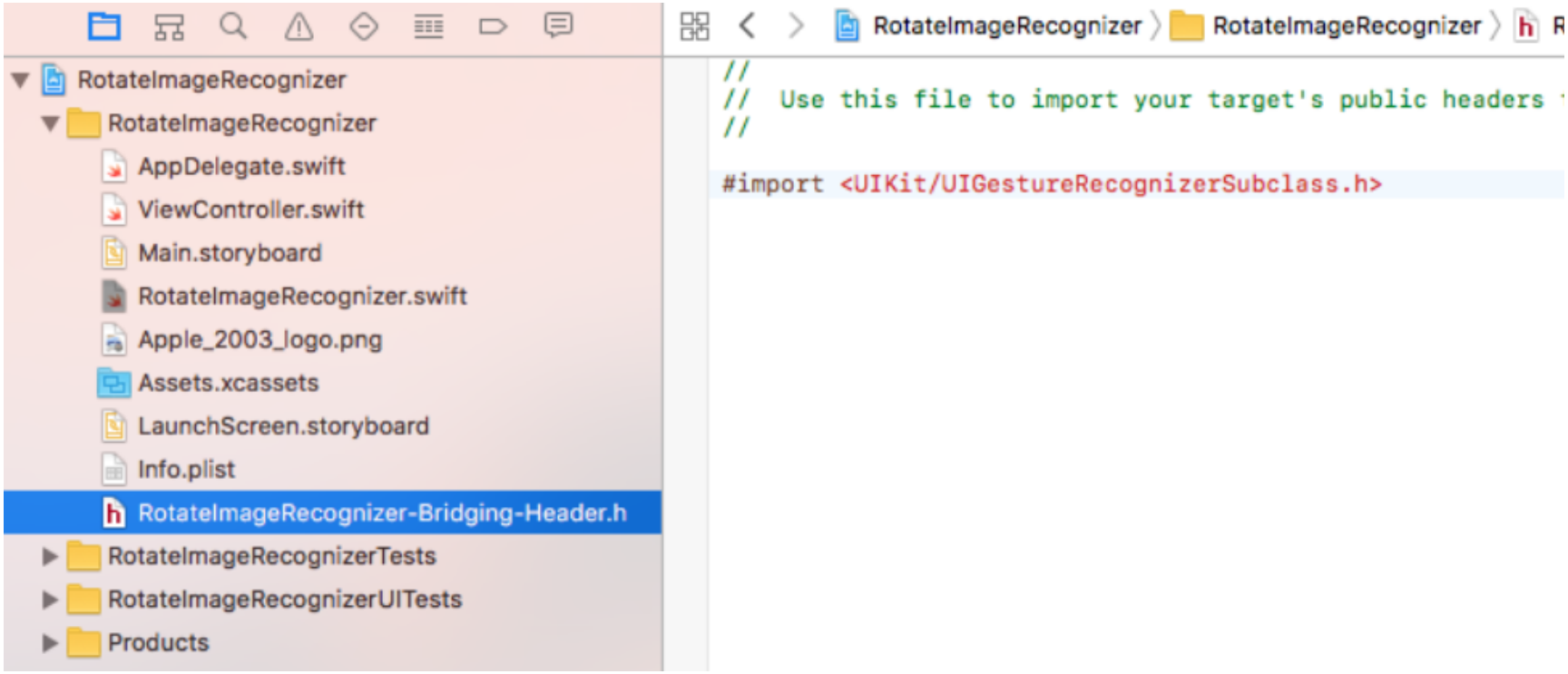
Vorgehensweise:

- Swift-Projekt anlegen
- Im Storyboard UIImageView hinzufügen und im Attributes Inspector Bild importieren
- Objective-C Klasse "UIGestureRecognizerSubclass" importieren
- Eigenen GestureRecognizer schreiben
- Im ViewController GestureRecognizer hinzufügen
- Action-Methode im ViewController programmieren

Importieren von UIGestureRecognizerSubclass

- nötig, damit Gesture Recognizer state Property ändern kann
- Objective-C Klasse "Bridging-Header" erstellen
- Nachfrage ob Bridging-Header erstellt werden soll mit ja beantworten
- "UIKit/UIGestureRecognizerSubclass.h" in entstandener .h-Klasse importieren
- "Bridging-Header.m" löschen

Importieren von UIGestureRecognizerSubclass



GestureRecognizer programmieren

- Klasse muss von "UIGestureRecognizer" erben
- Überschreibe "touchesBegan"
- Überschreibe "touchesMoved"
- Überschreibe "touchesEnded"
- Überschreibe "touchesCancelled"
- Überschreibe "reset"

GestureRecognizer programmieren

```
import Foundation
import UIKit

class RotateImageRecognizer:UIGestureRecognizer{

    //benoetigte Anzahl an Punkten, die man sich bei einer Geste bewegen muss
    let requiredPointsSideways = 25
    let requiredPointsDown = 25
    var startpoint : CGPoint = CGPoint.zero


    //enum ermittelt in welcher Phase der Geste man sich gerade befindet
    enum Direction:Int {
        case DirectionUnknown = 0
        case DirectionLeft
        case DirectionLeftDown
        case DirectionLeftDone
        case DirectionRight
        case DirectionRightDown
        case DirectionRightDone
    }


    var currentDirection:Direction = .DirectionUnknown


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

GestureRecognizer programmieren

```
override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent) {
    super.touchesBegan(touches, with: event)
    //Geste soll nur mit einem Finger funktionieren
    if touches.count != 1 {
        self.state = .failed;
        return
    }
    if let touch = touches.first{
        //speichere, wo Finger display getroffen hat
        self.startpoint = touch.location(in: self.view)
        self.state = .possible
    }


}

override func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent) {
```

GestureRecognizer programmieren (touchesMoved 1)

```
override func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent) {
    super.touchesMoved(touches, with: event)
    if self.state == .failed {
        return
    }
    if let touch = touches.first{
        let nowPoint = touch.location(in: self.view)
        let prevPoint = touch.previousLocation(in: self.view)
        //Finger darf sich nicht nach oben bewegen
        if nowPoint.y < prevPoint.y{
            print("Going up, break")
            self.state = .failed
            return
        }
        //Fallunterscheidung: darf sich Finger nur nach links oder nur nach rechts bewegen,
        switch self.currentDirection {
```

GestureRecognizer programmieren (touchesMoved 2)

```
,
//Fallunterscheidung: darf sich Finger nur nach links oder nur nach rechts bewegen,
switch self.currentDirection {
case .DirectionUnknown:
    if nowPoint.x > prevPoint.x {
        self.currentDirection = .DirectionRight
    }else if nowPoint.x < prevPoint.x{
        self.currentDirection = .DirectionLeft
    }
case .DirectionLeft:
    if nowPoint.x > prevPoint.x{
        self.state = .failed; return
    }else if self.startpoint.x - nowPoint.x >= 25{
        currentDirection = .DirectionLeftDown
    }
case .DirectionRight:
    if nowPoint.x < prevPoint.x{
        self.state = .failed; return
    }else if nowPoint.x - self.startpoint.x >= 25{
        currentDirection = .DirectionRightDown
    }
case .DirectionLeftDown:
    if nowPoint.x > prevPoint.x{
        self.state = .failed; return
    } else if nowPoint.y-self.startpoint.y >= 25{
        currentDirection = .DirectionLeftDone
    }
case .DirectionRightDown:
    if nowPoint.x < prevPoint.x{
        self.state = .failed; return
    }else if nowPoint.y-self.startpoint.y >= 25{
        currentDirection = .DirectionRightDone
    }
case .DirectionLeftDone:
    if nowPoint.x > prevPoint.x{
        self.state = .failed; return
    }
case .DirectionRightDone:
    if nowPoint.x < prevPoint.x{
        self.state = .failed; return
    }

}
```

GestureRecognizer programmieren

```
override func touchesEnded(_ touches: Set<UITouch>, with event: UIEvent) {
    super.touchesEnded(touches, with: event)
    if self.state == .possible{
        if let touch = touches.first{
            let nowPoint = touch.location(in: self.view)
            let prevPoint = touch.previousLocation(in: self.view)

            if self.currentDirection == .DirectionLeftDone &&
                    !(nowPoint.x > prevPoint.x || nowPoint.y < prevPoint.y){
                self.state = .recognized
            }else if self.currentDirection == .DirectionRightDone &&
                !(nowPoint.x < prevPoint.x || nowPoint.y < prevPoint.y){
                self.state = .recognized
            }else{
                self.state = .failed
            }
        }

    }

}
```

GestureRecognizer programmieren

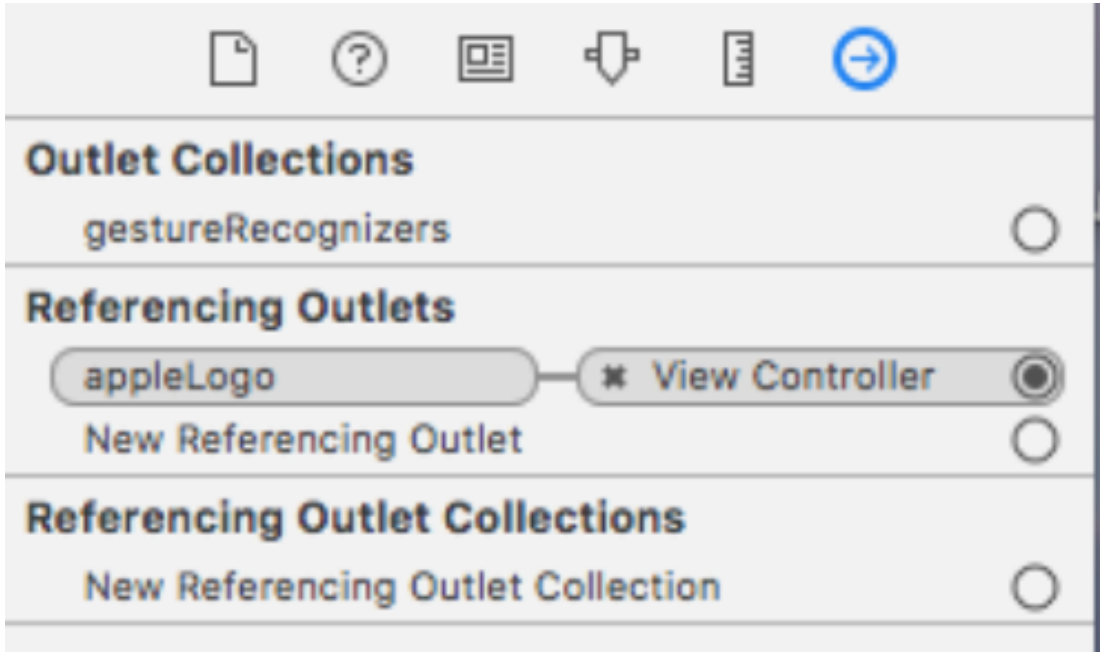
```
override func touchesCancelled(_ touches: Set<UITouch>, with event: UIEvent) {
    super.touchesCancelled(touches, with: event)
    currentDirection = .DirectionUnknown
    startpoint = CGPoint.zero
    self.state = .failed
}

override func reset() {
    super.reset()
    currentDirection = .DirectionUnknown
    startpoint = CGPoint.zero

}
```

ViewController

- Mit "ctrl-drag" Image in ViewController ziehen
- Connections-Inspektor überprüfen



ViewController

- GestureRecognizer in ViewDidLoad hinzufügen

```
class ViewController: UIViewController{

    @IBOutlet weak var appleLogo: UIImageView!
    var rotationAngle:Double = 0

    override func viewDidLoad() {
        //appleLogo = UIImageView(image: UIImage(named: "Apple_2003_logo.png"))

        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.


        let tgr = RotateImageRecognizer(target: self, action: #selector(self.handleRotateImage(tgr:)))
        self.view.addGestureRecognizer(tgr)


    }
```

ViewController

- Actionmethode programmieren

```
func handleRotateImage(tgr: RotateImageRecognizer) -> Void {
    print("Rotate")
    if tgr.currentDirection == .DirectionLeftDone{
        print("rotate left")
        rotationAngle = (rotationAngle+M_PI_2+M_PI)
        rotationAngle = rotationAngle.truncatingRemainder(dividingBy: (2*M_PI))
        self.appleLogo.transform = CGAffineTransform(rotationAngle: CGFloat(rotationAngle))

    }else if tgr.currentDirection == .DirectionRightDone{
        print("rotate right")
        rotationAngle = (rotationAngle+M_PI_2)
        rotationAngle = rotationAngle.truncatingRemainder(dividingBy: (2*M_PI))
        self.appleLogo.transform = CGAffineTransform(rotationAngle: CGFloat(rotationAngle))
    }
}
```

Endergebnis