LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

mobile and distributed systems group

Praktikum iOS-Entwicklung

Sommersemester 2017

Prof. Dr. Linnhoff-Popien

Lenz Belzner, Kyrill Schmid

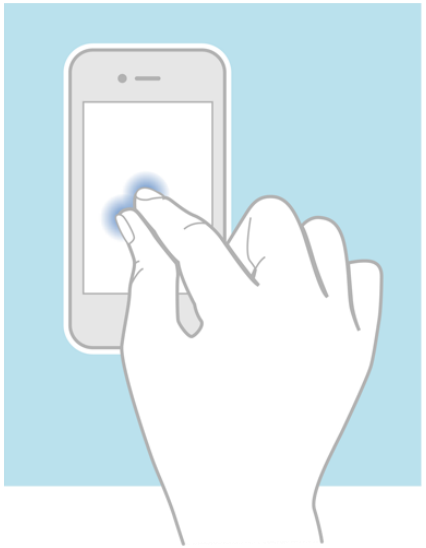
Event Handling – Interaktionen mit dem Display

Überblick

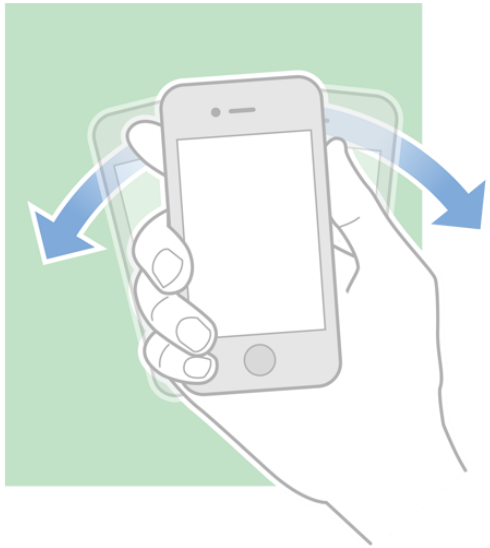
Der Nutzer kann durch Interaktion mit seinem Gerät verschiedene Events in iOS auslösen

Es gibt

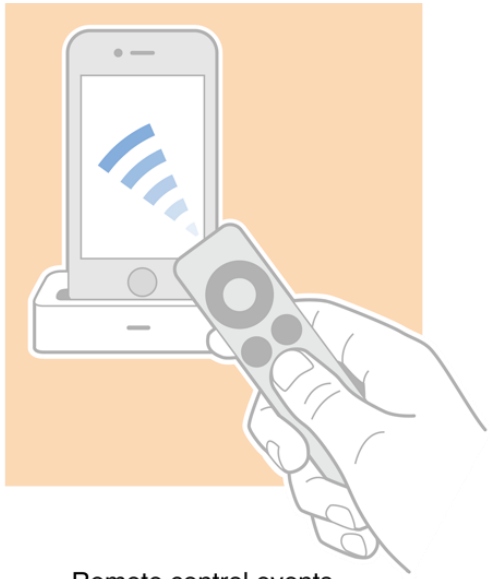
- (Multi-)Touch-Events
- Motion-Events
- Remote-Control-Events (zur Bedienung von Multimedia-Komponenten)



Multitouch events



Accelerometer events



Remote control events

Quelle: http://www.apple.com/

Überblick

Events sind Objekte, die an eine Anwendung gesendet werden, um sie über Interaktionen eines Nutzers zu informieren

Events, die durch den Nutzer ausgelöst werden, sind Instanzen der Klasse `UIEvent`

- enthalten Informationen, um auf das Event entsprechend zu reagieren
- haben einen Typ, z.B.
  - touch
  - shaking motion
  - remote control
- und einen Subtyp, z.B. für remote control
  - play
  - pause
  - stop
  - ...

Touch-Gesten

Touch-Gesten: Überblick

Behandlung einzelner Berührungen der Finger mit dem Display (**Touch-Events)** ist in vielen Fällen viel zu aufwendig

Viele Interaktionen erfolgen nach dem gleichen Schema (Gesten)

- Tippen (Tap)
- Wischen (Swipe)
- Klammern (Pinch)
- Lange Drücken (LongPress)
- ...

→ Verwendung einer "High-Level-API" zur Behandlung von Gesten

Touch-Gesten: Überblick

Geste = Kombinationen/Abfolgen von Touch-Events

Intuitive Gesten werden von iOS über "Gesture Recognition" direkt erkannt

- Tippen (`UITapGestureRecognizer`)
- Pinchen (`UIPinchGestureRecognizer`)
- Wischen (`UISwipeGestureRecognizer`)
- Ziehen/Verschieben (`UIPanGestureRecognizer`)
- ...

Für viele Gesten gibt es in Kombination mit typischen UI-Elementen eine direkte Integration in `UIKit`:

- Beispiel: `UIScrollView`
  - Pinch-Geste → Zoom
  - Wisch-Geste → Scroll

Touch-Gesten: Erkennen und behandeln

Gesture Recognizer (`UIGestureRecognizer`)

- Dienen der Abstraktion von der komplexen Event-Handling-Logik
- Stellen den bevorzugten Weg zur Behandlung von Touch-Events dar

Implementieren eigener Action-Target-Mechanismen:

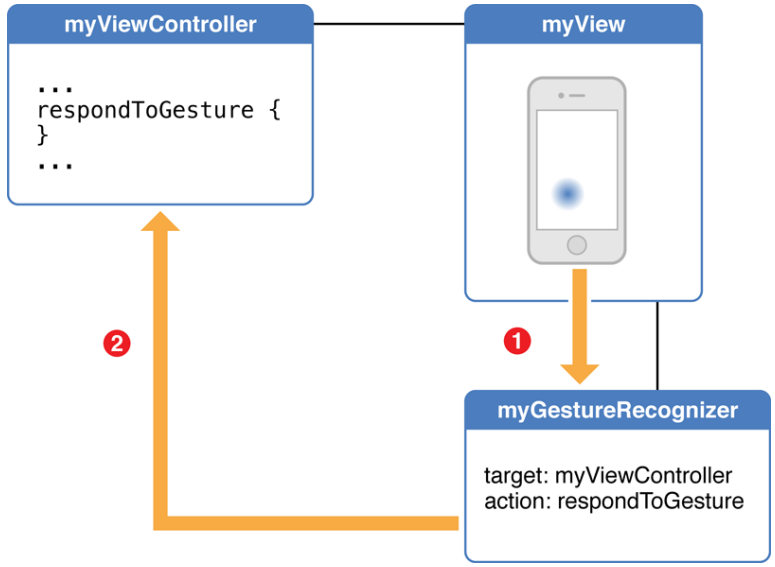
- Instanziieren und Konfigurieren eines Gesture Recognizers
- Hinzufügen des Gesture Recognizer zur eigenen View
  - Gesamte View reagiert auf die hinzugefügte Geste

Vorgehensweise:

- Verwendung und Anpassung von **Built-In** Gesture Recognizern, oder
- Implementierung eigener (**Custom**) Gesture Recognizer

Hinzufügen eines Built-In Gesture Recognizers

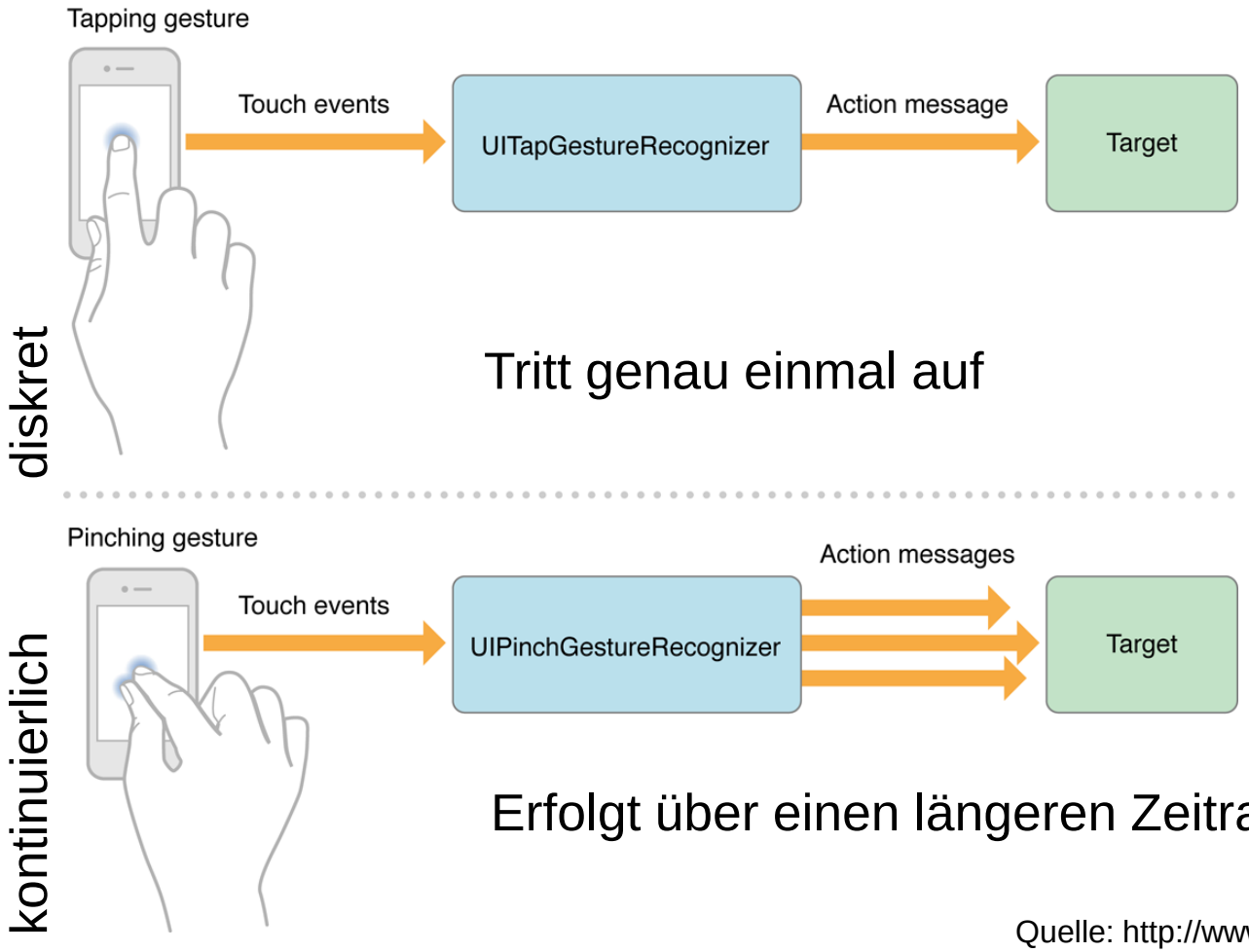
- Erzeugen und konfigurieren einer Instanz von `UIGestureRecognizer`
  - Zuweisen eines Targets und einer Action
  - Zuweisen gestenspezifischer Attribute (optional; z.B. `numberOfTapsRequired`)
- Registrieren des Gesture Recognizers mit einer View
- Implementieren der Action-Methode



Quelle: http://www.apple.com/

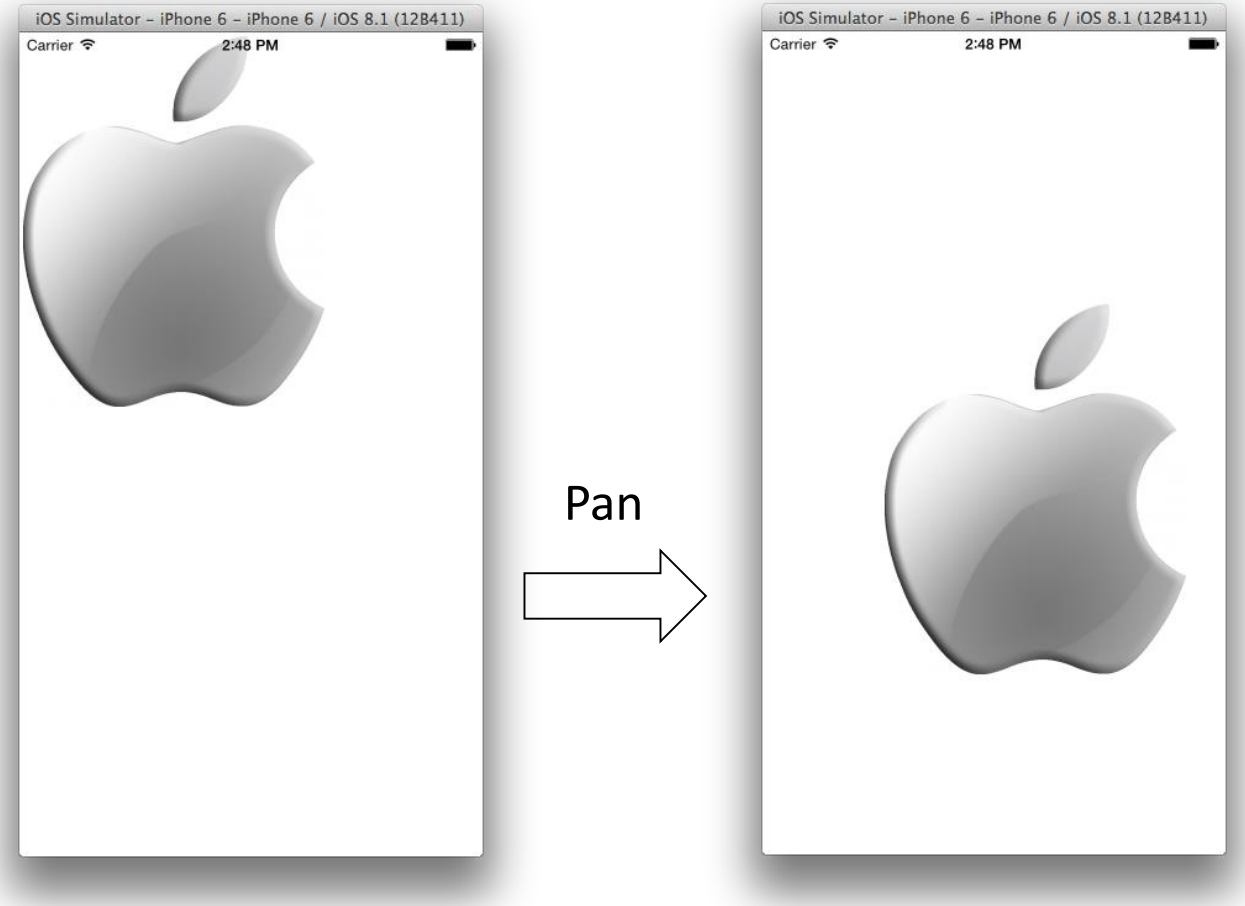
Unterscheidung von Gesten

Es existieren **diskrete** und **kontinuierliche** Gesten



Quelle: http://www.apple.com/

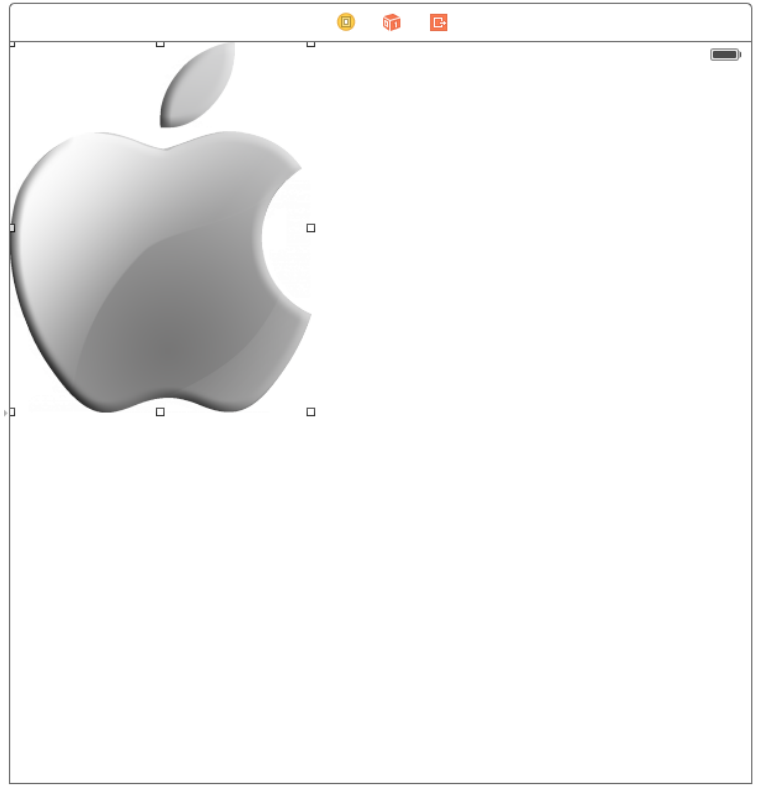
Beispiel: ImageTranslationRecognition (Storyboard)



Beispiel: ImageTranslationRecognition

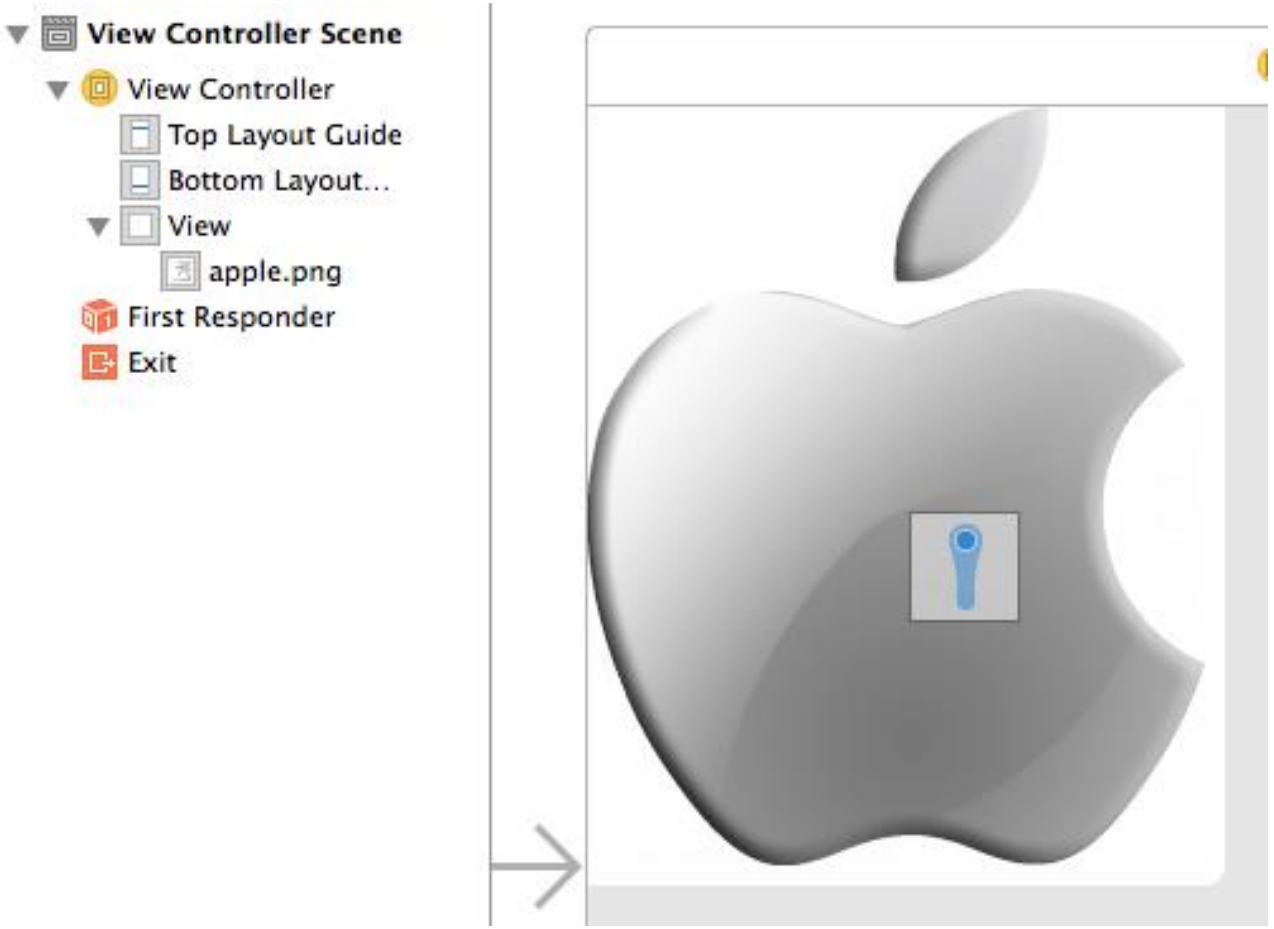
Vorgehensweise

- Erzeugen eines neuen Projekts "ImageTranslationRecognition" (Template: Single View Application)
- Im Storyboard:
  - Hinzufügen einer Image View vom Typ `UIImageView`
- Im Project Navigator:
  - Hinzufügen eines Image (Drag & Drop vom Finder)
- Unter „Editor“:
  - Size to fit content: Anpassen der Größe der Image View an das Image



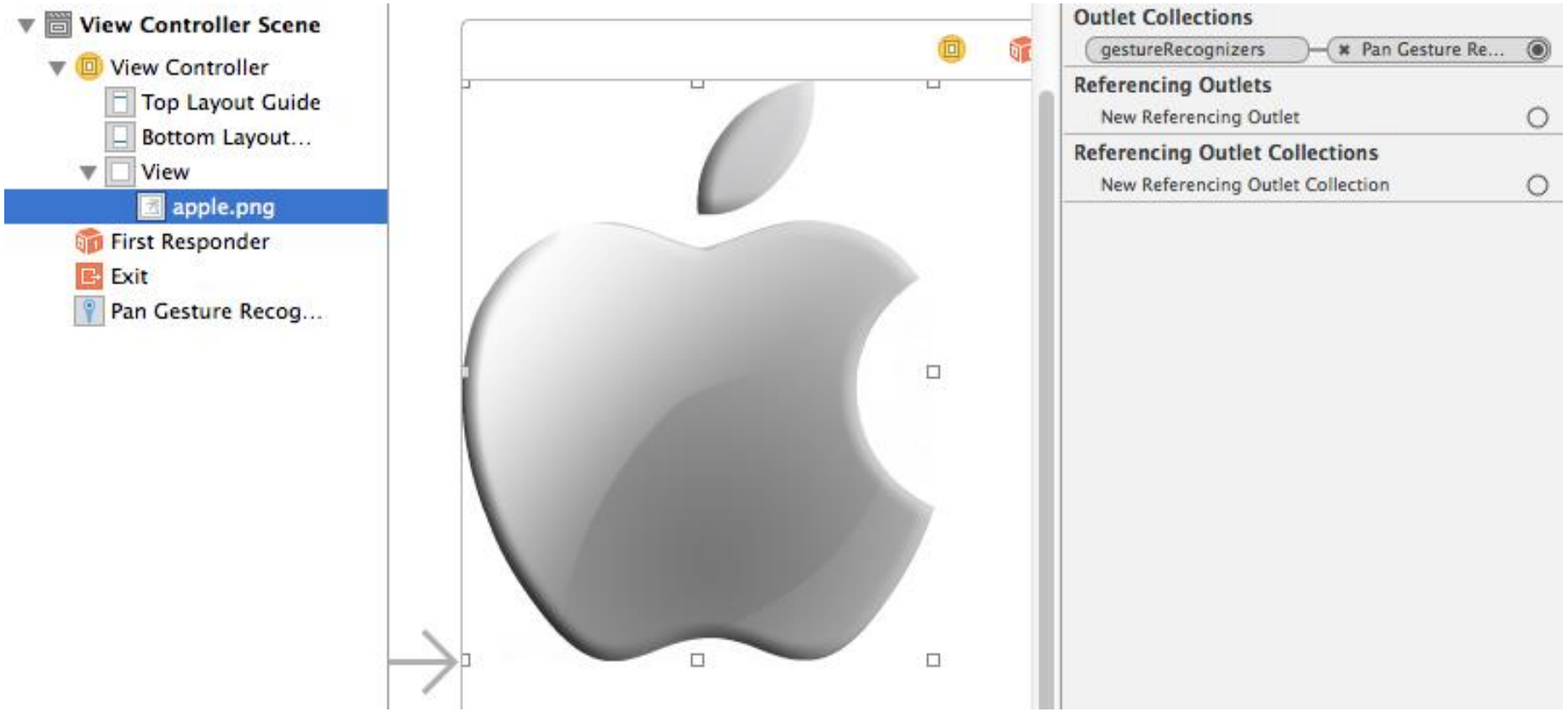
Beispiel: ImageTranslationRecognition

- Von der Objekt Library:
  - Hinzufügen eines Pan Gesture Recognizers vom Typ `UIPanGestureRecognizer` zur Image View (Drag & Drop auf die View)



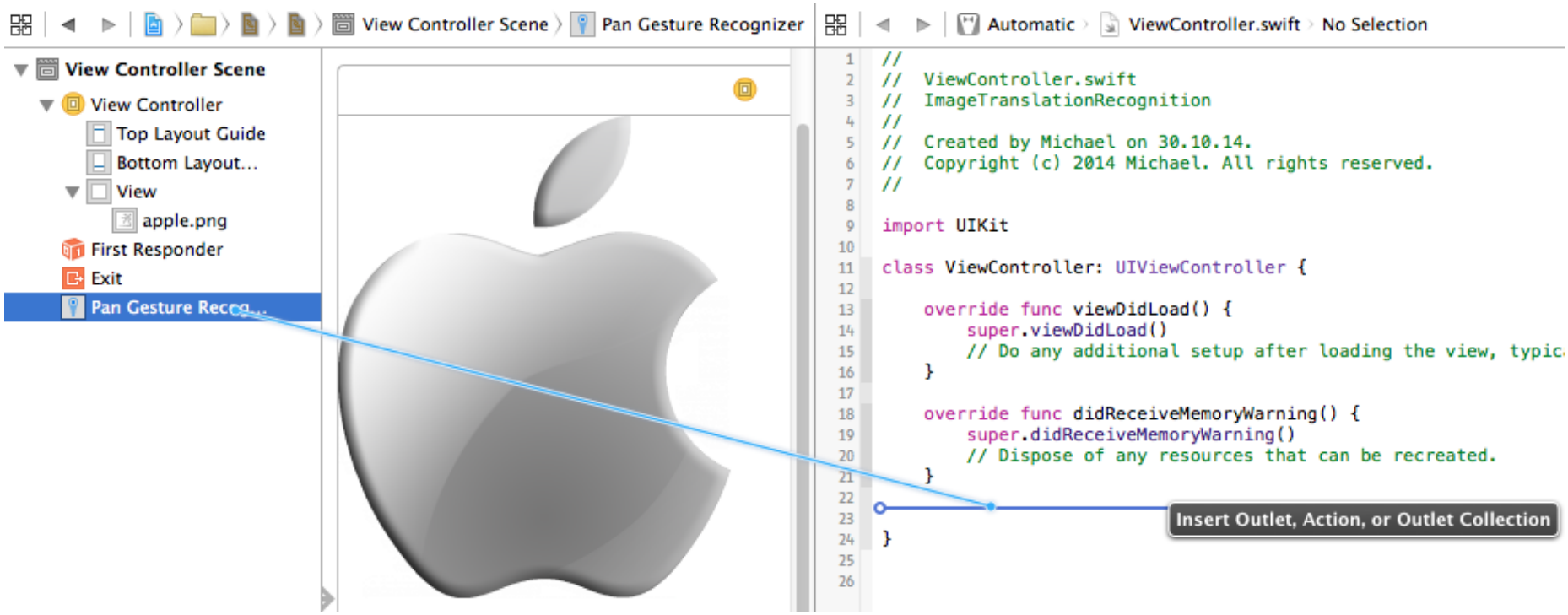
Beispiel: ImageTranslationRecognition

- Pan Gesture Recognizer ist jetzt als Outlet bei der Image View registriert



Beispiel: ImageTranslationRecognition

- Erzeugen und Verbinden einer Action-Methode (Controller ist Target)



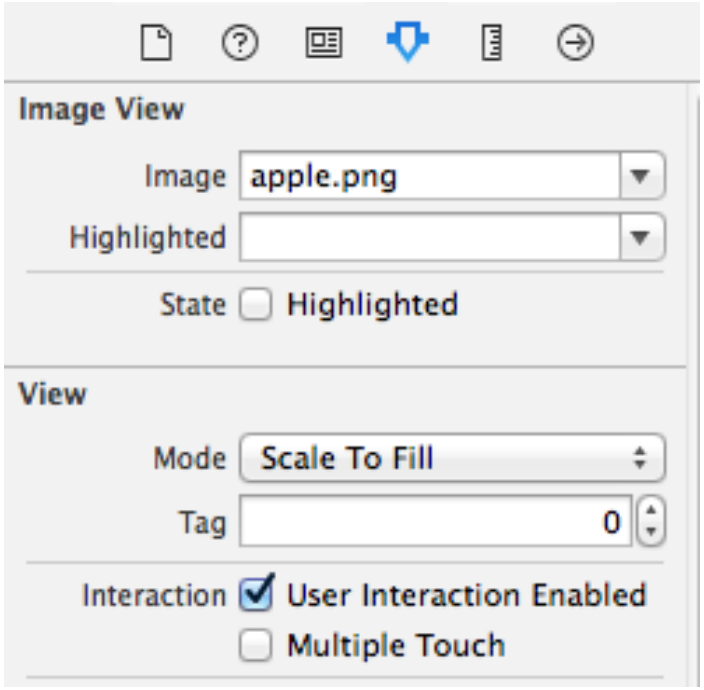
Beispiel: ImageTranslationRecognition

- Implementierung der Action-Methode

```
//  ViewController.swift

import UIKit

class ViewController: UIViewController {

    @IBAction func handlePan(sender: UIPanGestureRecognizer) {
        // Die Translation liefert hier den absoluten Abstand
        // zu dem Punkt, an dem der Finger auf den Bildschirm traf
        // bezogen auf das Koordinatensystem der übergebenen View
        let translation:CGPoint = sender.translationInView(self.view)

        // Anpassen der Position des Images
        sender.view!.center = CGPoint(x:sender.view!.center.x + translation.x,
            y:sender.view!.center.y + translation.y)

        // Zurücksetzen der Translation, um nur relative Positionsveränderungen
        // zur Position im vorhergehenden Aufruf der Action-Methode zu erhalten
        sender.setTranslation(CGPoint.zero, inView: self.view)
    }
}
```

Beispiel: ImageTranslationRecognition

Im Attributes Inspector: Erlauben von Nutzerinteraktionen mit der Image View



Beispiel: ImageTranslationRecognition

Allgemeine Anmerkungen

- Diskrete Gesten rufen Action Methode einmal auf
- Kontinuierliche Geste kann Action Methode mehrfach aufrufen
- Das Referenzieren eines Gesture Recognizer außerhalb der Action-Methode ist über eine Outlet-Connection möglich

Anmerkungen zum `UIPinchGestureRecognizer`

- Wenn man den Skalierungsfaktor nicht selbst verändert, wird er absolut berechnet
  - s Skalierungsfaktor
  - d_{Start} Abstand der Finger bei erster Berührung
  - d_{Now} Gegenwärtiger Abstand der Finger

→ $s = d_{Now} / d_{Start}$

Programmatisches Erzeugen eines UIGestureRecognizers

Initialisierung mit `initWithTarget:action:`

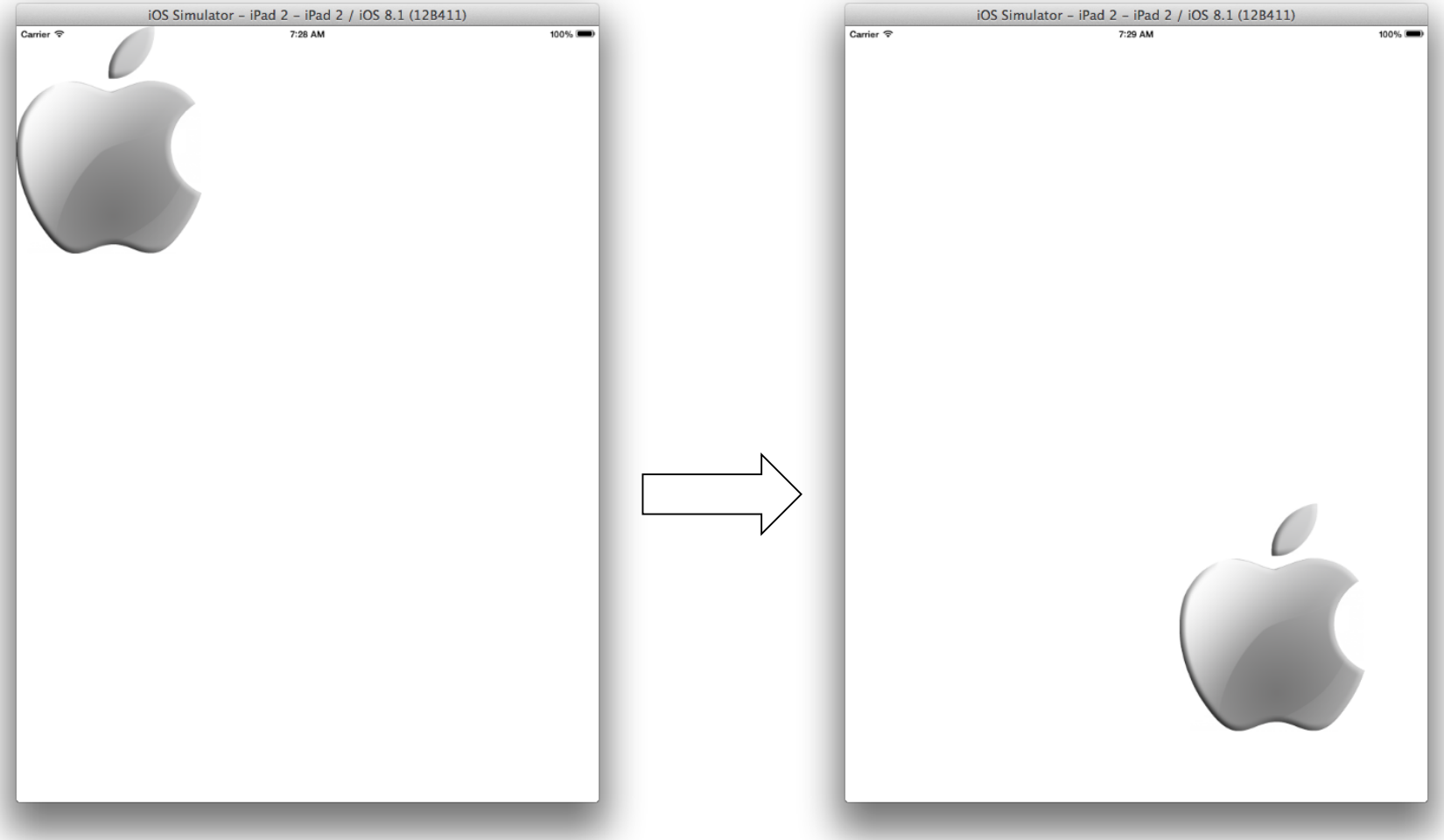
- `target`:
  - Target-Objekt, das in einer Action-Methode auf eine Geste reagiert
  - I.d.R. der zugehörige View Controller
- `action`:
  - Ein **Selektor**
  - Spezifiziert die Action-Methode, die beim Auftreten der Geste ausgeführt werden soll

Registrieren des Gesture Recognizers mit `addGestureRecognizer:` des entsprechenden `UIView`-Objekts

Programmatische Erzeugung erfolgt i.d.R. in `viewDidLoad`

Beispiel: TapRecognition (programmatisch)

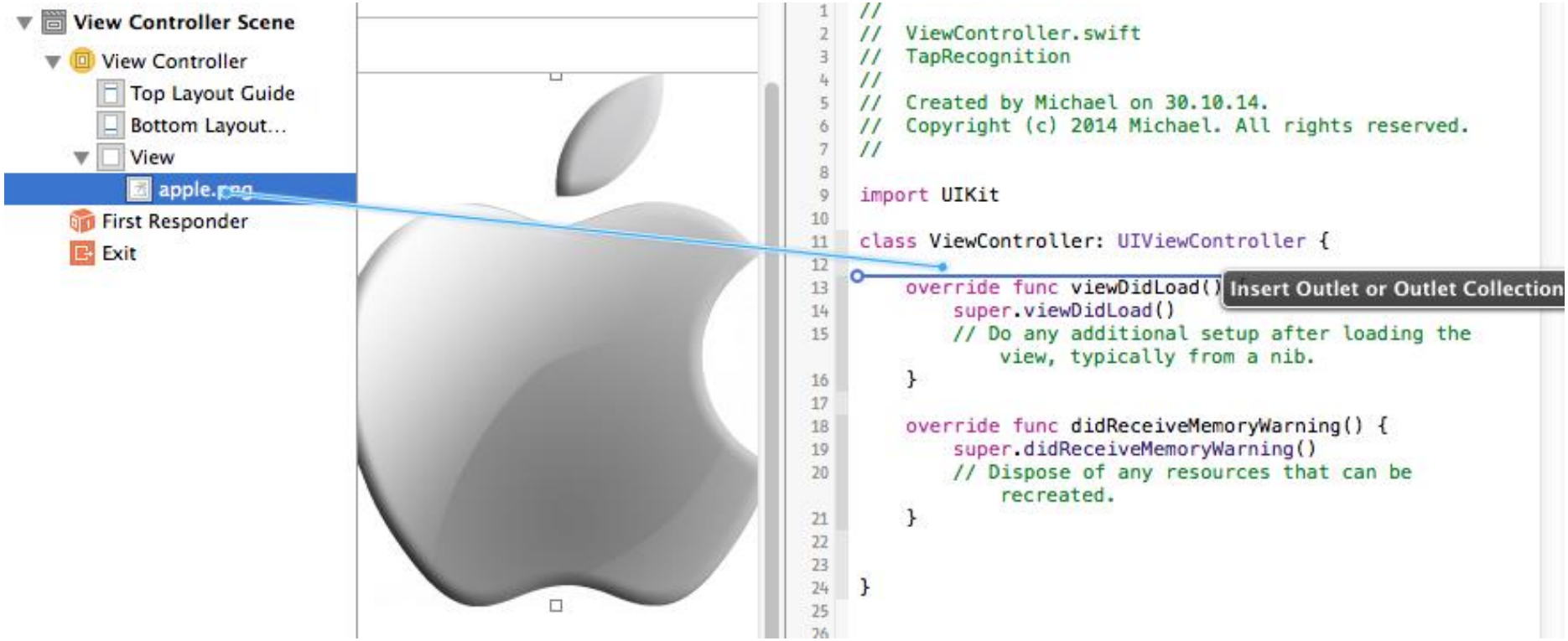
Tippen auf den Bildschirm bewirkt das Verschieben eines Images an die angetippte Stelle



Beispiel: TapRecognition

Vorgehensweise

- Analog zum Beispiel ImageTranslationRecognition
- Erzeugen einer Outlet-Connection zur Image View




```swift
//
//  ViewController.swift
//  TapRecognition
//
//  Created by Michael on 30.10.14.
//  Copyright (c) 2014 Michael. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

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

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be
            recreated.
    }


}
```

Beispiel: TapRecognition

Programmatisches Hinzufügen eines `UITapGestureRecognizers` (geht natürlich auch über Interface-Builder...)

```
//  ViewController.swift

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var imageView: UIImageView!

    override func viewDidLoad() {
        // Initialisieren eines UITapGestureRecognizers
        let tgr = UITapGestureRecognizer(target: self,
                    action:#selector(handleTap))

        // Registrieren mit der eigenen View
        self.view.addGestureRecognizer(tgr)
    }
}
```

Beispiel: TapRecognition

Implementierung der Action-Methode

- Achtung: Nur zwei Signaturen erlaubt:
  - `func handleTap() -> Void`
  - `func handleTap(tgr:UITapGestureRecognizer) -> Void`

```
func handleTap(tgr:UITapGestureRecognizer) -> Void {
    // Position wo das Event auftrat
    let location = tgr.locationInView(self.view)

    // Verschieben des Images an die neue Position
    self.imageView.center = location;

}
```

Eine Instanz von `UIGestureRecognizer` gehört **zu genau einer** `UIView`-Instanz

- Hinzufügen desselben Gesture Recognizers zu verschiedenen `UIView`-Instanzen nicht möglich!
- Ein wiederholtes Hinzufügen überschreibt das vorhergehende!
- Beispiel:
  - Nur `view2` erhält die Nachricht `tapTapTap`

```
// MyController.m
[...]

UITapGestureRecognizer *tapGesture =
   [[UITapGestureRecognizer alloc] initWithTarget:self
      action:@selector(tapTapTap:)];

[self.view1 addGestureRecognizer:tapGesture];
[self.view2 addGestureRecognizer:tapGesture];

[...]
```

Behandlung von Touch-Gesten im Gesture Recognizer

`UIGestureRecognizer` unterbricht standardmäßig Behandlung von Touch-Events, wenn eine Geste erkannt wurde

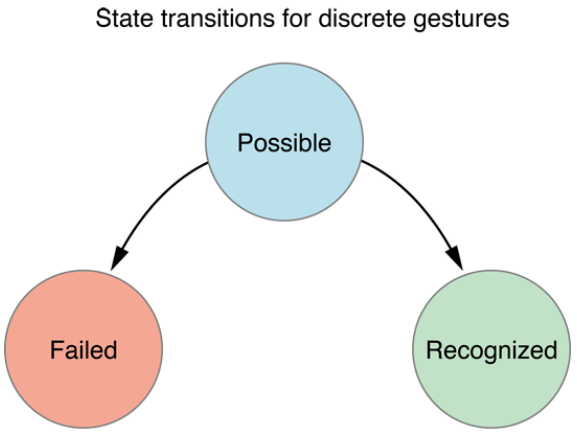
Problem:

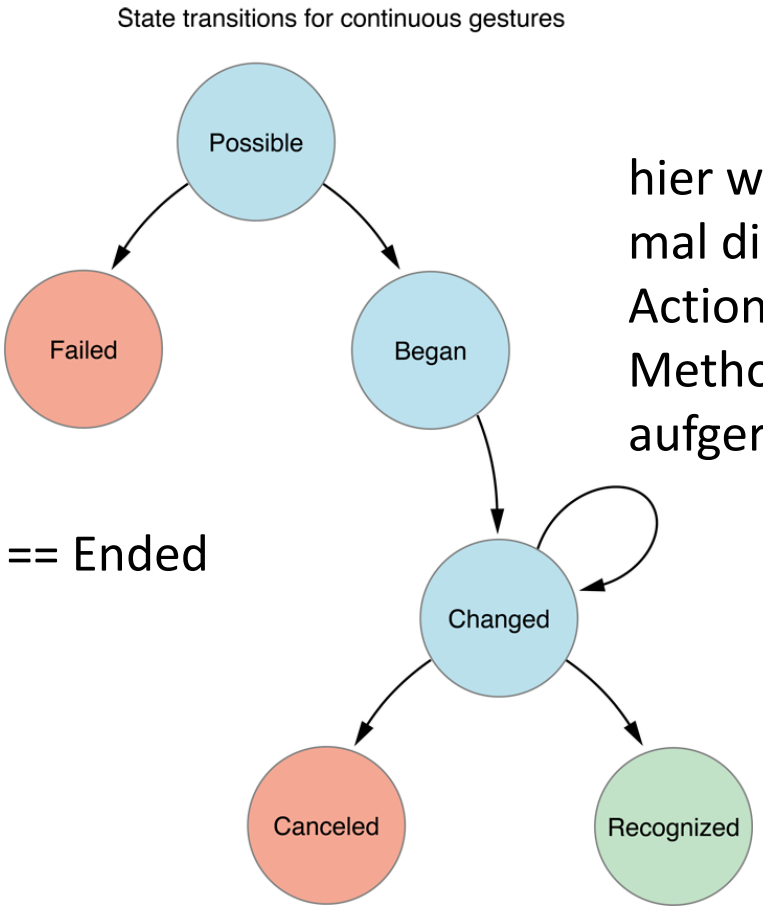
- `UIView` mit einer Gestenerkennung erhält möglicherweise nicht alle Touch-Events.

Gesture Recognizer: Zustandsmodell

Gesture Recognizer operieren in einer Zustandsmaschine

- mehr dazu im *"Event Handling Guide for iOS"*





hier wird jedes mal die Action-Methode aufgerufen

Achtung: Recognized == Ended

Quelle: http://www.apple.com/

Gesture Recognizer: Zustandsmodell

Testen des gegenwärtigen Zustands über die Konstanten

- `UIGestureRecognizerStatePossible` (Default State)
  - Es wurden eventuell Touch-Events registriert, aber noch keine Geste erkannt
- `UIGestureRecognizerStateBegan`
  - Es wurde ein Touch-Event empfangen und eine **kontinuierliche** Geste erkannt
    → Löst eine Action-Nachricht aus
- `UIGestureRecognizerStateChanged`
  - Es wurden Touch-Events erkannt, die eine Veränderung in der **kontinuierlichen** Geste signalisieren
    → Löst eine Action-Nachricht aus
- `UIGestureRecognizerStateEnded`
  - Es wurden Touch-Events erkannt, die das Ende einer (**kontinuierlichen**) Geste signalisieren
    → Löst eine Action-Nachricht aus
    → Reset auf `UIGestureRecognizerStatePossible`

Gesture Recognizer: Zustandsmodell

Testen des gegenwärtigen Zustands über die Konstanten

- `UIGestureRecognizerStateFailed`
  - Es wurde eine Sequenz von Touch-Events empfangen, die aber nicht als Teil der gegenwärtigen Geste erkannt werden konnten
    → Keine Action-Nachricht
    → Reset auf `UIGestureRecognizerStatePossible`

- `UIGestureRecognizerStateRecognized`
  - Es wurden eine Sequenz von Touch-Events als Geste erkannt
    → Löst eine Action-Nachricht aus
    → Reset auf `UIGestureRecognizerStatePossible`

Hinweise:

- Die Konstanten sind als `enum` realisiert.
- In Swift erfolgt der Zugriff jedoch über die Member Values (z.B. `UIGestureRecognizerState.Began`)
- In Swift existiert keine Konstante `.Recognized`

Gesture Recognizer: Erkennen paralleler Gesten

Beispiel: Drag & Drop:

- Realisierung normalerweise mit LongPress und Pan auf ein UI-Objekt.
- LongPress zum Starten des Drag-Vorgangs, Pan zum Bewegen
- Die gesamte Pan-Geste **liegt vollständig innerhalb** der LongPress-Geste!

Problem:

- Standardmäßig wird das parallele Erkennen mehrerer Gesten unterdrückt
- Hier muss aber `Pan` während eines `LongPress` erkannt werden
- LongPress gibt Touch-Events nicht weiter
  → Pan kann nicht erkannt werden!

Lösung:

- Implementierung des `UIGestureRecognizerDelegate` Protokolls

Protokoll: Was ist das?

Die Klasse eines Objekts unterscheidet sich häufig von dessen **Rolle** in einem System.

Beispiel

- Die Klasse eines Objekts ist `NSMutableArray` aber seine Rolle in der Anwendung ist eine Warteschleife (für Druckaufträge).

Spezifikation von Rollen erfolgt in Objective-C / Swift über Protokolle (ähnlich zu Interfaces in Java).

Protokolle

- stellen eine Alternative zur Vererbung dar
- können von beliebigen Klassen implementiert werden
- ermöglichen es Objekten schwach assoziierter Klassen miteinander zu kommunizieren

Protokoll: Definition in Objective C

Protokolle spezifizieren die zu implementierenden Methoden

Methoden können als **verpflichtend** oder **optional** spezifiziert werden

Beispiel: `UIGestureRecognizerDelegate`-Protokoll

```
// Dieses Protokoll ist konform zum NSObject Protokoll
@protocol UIGestureRecognizerDelegate <NSObject>

// die folgenden Methoden MÜSSEN implementiert werden
@required
[...]

// die folgenden Methoden KÖNNEN implementiert werden
@optional
-(BOOL)gestureRecognizer:(UIGestureRecognizer *)gr
    shouldRecognizeSimultaneouslyWithGestureRecognizer:
    (UIGestureRecognizer *)o;

[...]

@end
```

Protokoll: Definition in Swift

Protokolle können ebenfalls als **verpflichtend** oder **optional** spezifiziert werden

```
protocol SomeProtocol {

    // die folgenden Methoden MÜSSEN implementiert werden
    func doSomeCalculatiom(count: Int) -> Int

    // die folgenden Methoden KÖNNEN implementiert werden
    optional func doSomeOtherStuff() -> Void

}
```

In Swift existieren viele weitere Möglichkeiten zur Definition von Protokollen und zur Restriktion ihrer Anwendung

- Siehe: https://developer.apple.com/library/ios/documentation/swift/conceptual/Swift_Programming_Language/Protocols.html

Gesture Recognizer: Erkennen paralleler Gesten

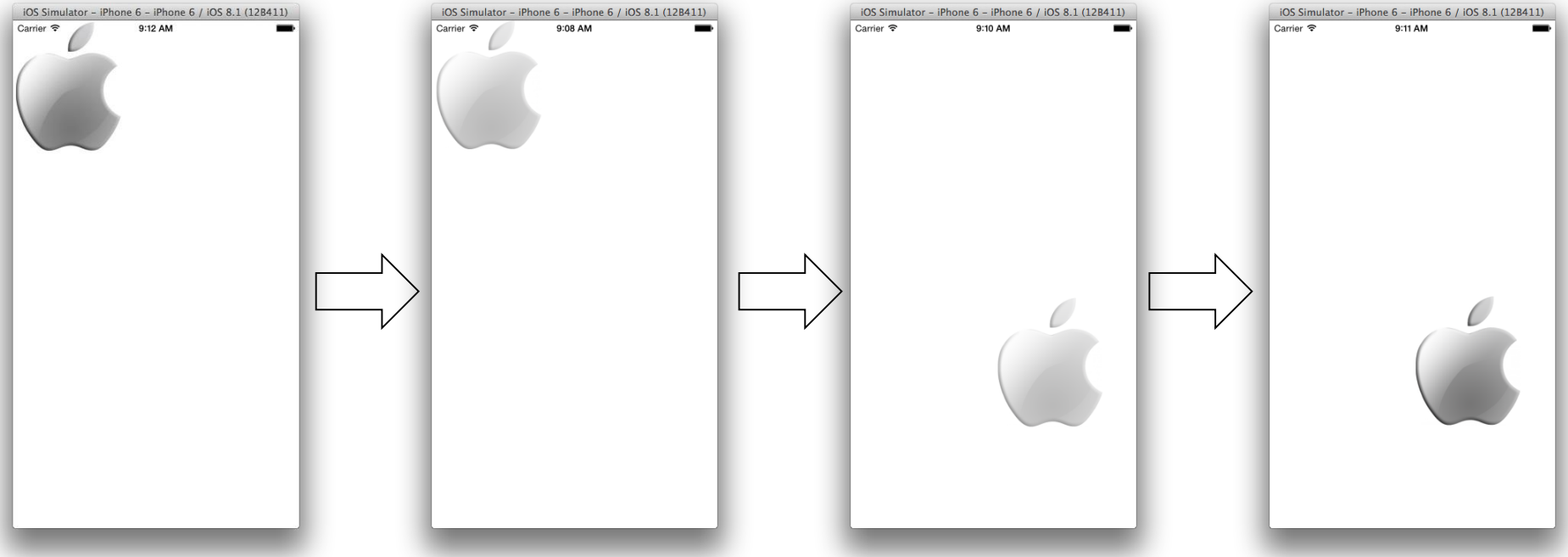
Problem:

- Parallele Erkennung von LongPress und Pan

Lösung:

- Implementierung des UIGestureRecognizerDelegate Protokolls

Beispiel: DragAndDrop



Long Press Pan Drop

Beispiel: DragAndDrop

Vorgehensweise

- Analog zum Beispiel "ImageTranslationRecognition"
- Kennzeichnung des View Controller als Delegate

```
//  ViewController.swift

import UIKit

class ViewController: UIViewController, UIGestureRecognizerDelegate {
    [...]
}
```

Beispiel: DragAndDrop

Erzeugen notwendiger Outlets und Properties

```
//  ViewController.swift

import UIKit

class ViewController: UIViewController, UIGestureRecognizerDelegate {

    @IBOutlet weak var imageView: UIImageView!
    var panGR:UIPanGestureRecognizer!
    var longPressGR:UILongPressGestureRecognizer!

    [...]
}
```

Initialisierung der Gesture Recognizer

```
//  ViewController.swift

import UIKit

class ViewController: UIViewController, UIGestureRecognizerDelegate {
    [...]

    override func viewDidLoad() {
        // Initialisieren/Registrieren eines UILongPressGestureRecognizer
        self.longPressGR = UILongPressGestureRecognizer(
            target:self, action:"handleLongPress:")
        self.imageView.addGestureRecognizer(self.longPressGR)

        // Initialisieren/Registrieren eines UIPanGestureRecognizer
        self.panGR = UIPanGestureRecognizer(target: self, action: "handlePan:")
        self.imageView.addGestureRecognizer(self.panGR)

        // Setzen des UILongPressGestureRecognizer als Delegate
        self.longPressGR.delegate = self
    }
}
```

Beispiel: DragAndDrop

Gibt die Protokoll-Methode `true` zurück, können die beiden übergebenen `UIGestureRecognizer` parallel zueinander Gesten erkennen.

```
//  ViewController.swift

import UIKit

class ViewController: UIViewController, UIGestureRecognizerDelegate {
    [...]

    func gestureRecognizer(
            UIGestureRecognizer,
            shouldRecognizeSimultaneouslyWithGestureRecognizer:
            UIGestureRecognizer) -> Bool {
        // Erlaube parallele Verarbeitung von Pan während LongPress
        // Achtung: Dies erlaubt auch den LongPress während Pan!
        return true
    }

}
```

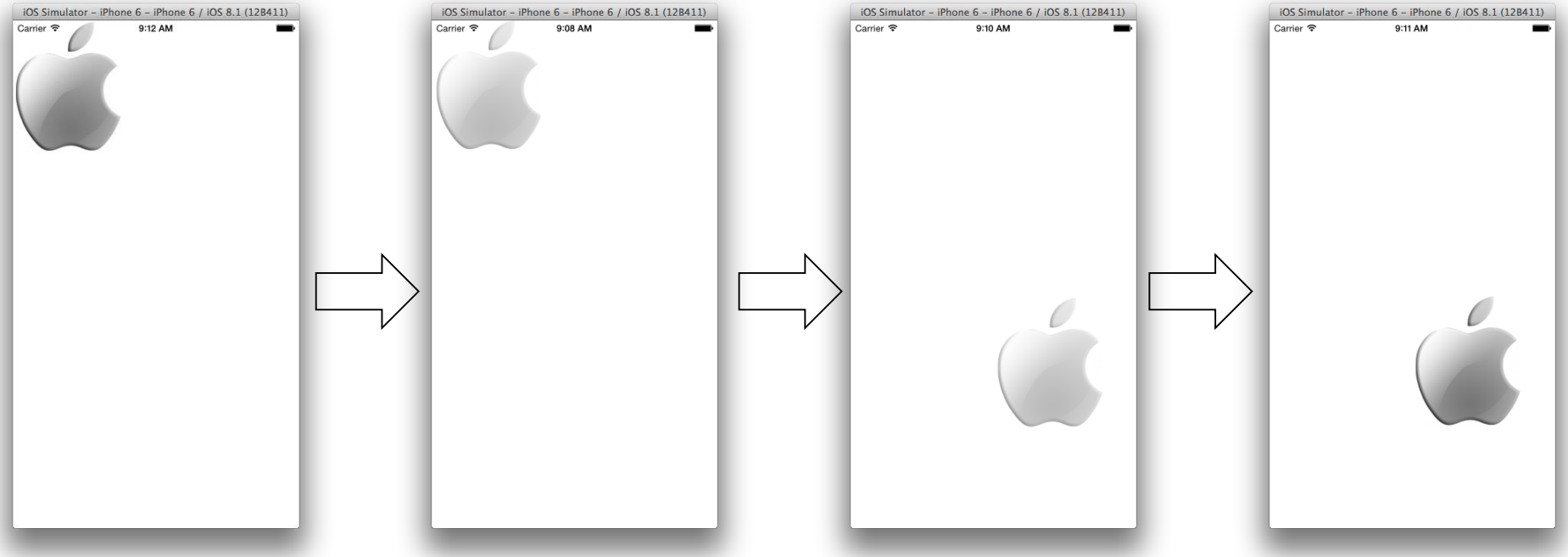
Implementierung des LongPress-Handlers

```
//  ViewController.swift

import UIKit

class ViewController: UIViewController, UIGestureRecognizerDelegate {
    [...]
    func handleLongPress(gestureRecognizer:UILongPressGestureRecognizer) {
        // Visualisiere, dass jetzt ein Drag & Drop passiert
        if gestureRecognizer.state == UIGestureRecognizerState.began
        || gestureRecognizer.state == UIGestureRecognizerState.Changed {
            self.imageView.alpha = 0.5
        }
        else {
            // Visualisiere, dass Drag & Drop zu Ende ist
            self.imageView.alpha = 1.0
        }
    }
}
```

Implementierung des Pan-Handlers

```
//  ViewController.swift

import UIKit

class ViewController: UIViewController, UIGestureRecognizerDelegate {

    [...]

    func handlePan(gestureRecognizer:UIPanGestureRecognizer) {
        // Führe Pan nur aus, wenn gerade ein LongPress aktiv ist
        if self.longPressGR.state == UIGestureRecognizerState.Began
        || self.longPressGR.state == UIGestureRecognizerState.Changed {
            // Verschiebe Image an die Position, wo das Event auftrat
            self.imageView.center =
                gestureRecognizer.locationInView(self.view)
        }
    }
}
```

Beispiel: DragAndDrop



Long Press Pan Drop