PG5600

iOS programmering

Forelesning 6

Praktisk informasjon

Skriftlig prøve og innlevering

Sist gang

- Delegate pattern
- UINavigationController
- UITableView og UITableViewController
- UICollectionsView og UICollectionViewController
- Auto Layout
- Unified Storyboard og Size classes

Agenda

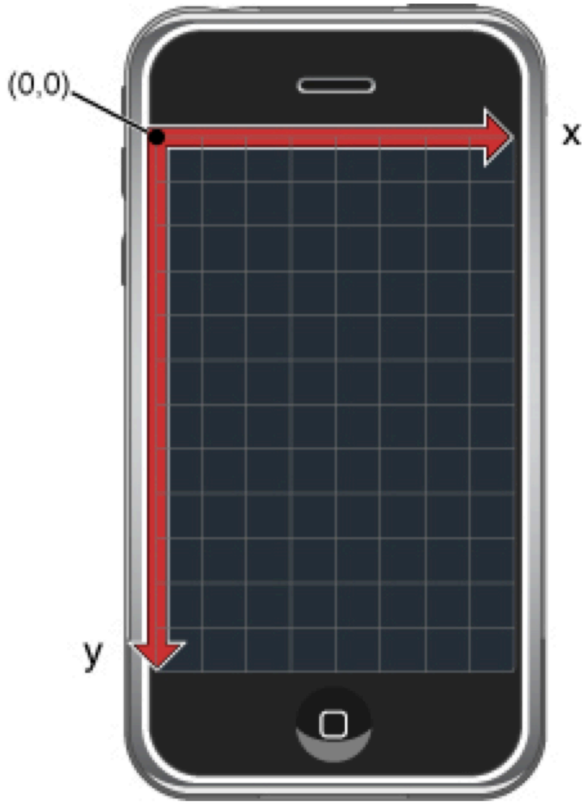
- Viewkonsepter
- Å instansiere views
- Å lage custom views
- Eventhåndtering
- Gestures
- Animasjoner

Views

- Hiarki med rektangulære views
  - subviews og superviews
- Kan håndterer events
  - Har typisk logikken hvor å definere hva et event betyr (eks. tap) og trigge dette
  - Men definerer ikke forretningslogikken som skal utføres (delegerer/kaller view controller i stedet)

Koordinatsystem

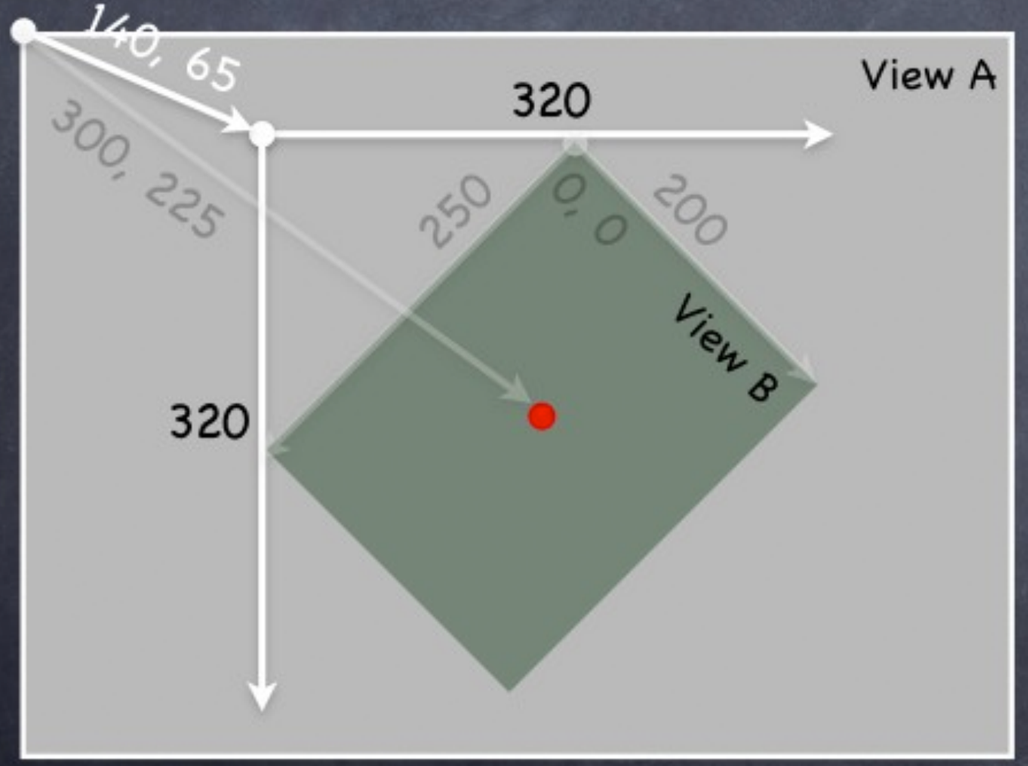
Figure 1–4 Coordinate system orientation in UIKit



Coordinates

- Use `frame` and `center` to position the view in the hierarchy

  These are used by `superviews`, never inside your `UIView` subclass's implementation.
  You might think `frame.size` is always equal to `bounds.size`, but you'd be wrong ...



Because views can be rotated
(and scaled and translated too).

View B's `bounds` = `((0,0),(200,250))`

View B's `frame` = `((140,65),(320,320))`

View B's `center` = `(300,225)`

View B's middle in its own coordinate space is
`(bound.size.width/2+bounds.origin.x,`
`bounds.size.height/2+bounds.origin.y)`
which is `(100,125)` in this case.

Views are rarely rotated, but don't
misuse `frame` or `center` by assuming that.

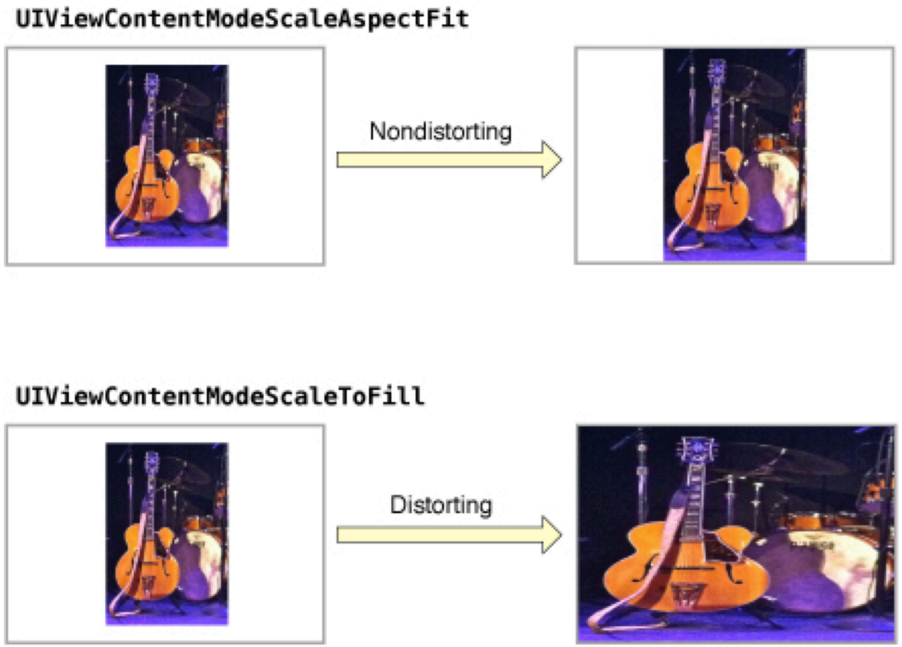
Kilde: Stanford CS193p
http://web.stanford.edu/class/cs193p

Frame vs. bounds

- Normalt bounds ved implementering av view (inside-out)
  - **bounds**: størrelse i eget koordinatsystem
- Normalt frame ved bruk av et view (outside-in)
  - **frame**: størrelse og posisjon i superview
  - **center**: senterposisjon ift. superview
- Oppdatering av frame/bounds/center vil oppdatere verdier i hverandre

Content mode

- Bestemmer hvordan et view sitt innhold tegnes når et view sine bounds forandres:



Transforms

```
func degrees2radians(degrees: Double) -> CGFloat {
    return CGFloat(M_PI * degrees / 180.0)
}

let rotate = CGAffineTransformMakeRotation(degrees2radians(90))
let scale = CGAffineTransformMakeScale(2, 2)
label.transform = CGAffineTransformConcat(scale, rotate)
```

Datastrukturer

```
let frame = CGRectMake(0, 0, 100, 100)
frame.origin // <-- CGPoint
frame.size // <-- CGSize

let point = CGPointMake(350, 22)
point.x
point.y

let size = CGSizeMake(200, 400)
size.width
size.height
```

```
{x 0 y 0 w 100 h 100}
{x 0 y 0}
{w 100 h 100}

{x 350 y 22}
350.0
22.0

{w 200 h 400}
200.0
400.0
```

Å instansiere views

Å instansiere views

- Fra Interface Builder (drag'n'drop)
- Fra kode

Fra kode

```
// I loadView: (evt. viewDidLoad:)
let label = UILabel(frame: CGRectMake(0, 0, view.frame.width, 20))
label.text = "Hello world"
label.textAlignment = NSTextAlignment.Center
view.addSubview(label)
```

Å lage custom views

Å lage custom views

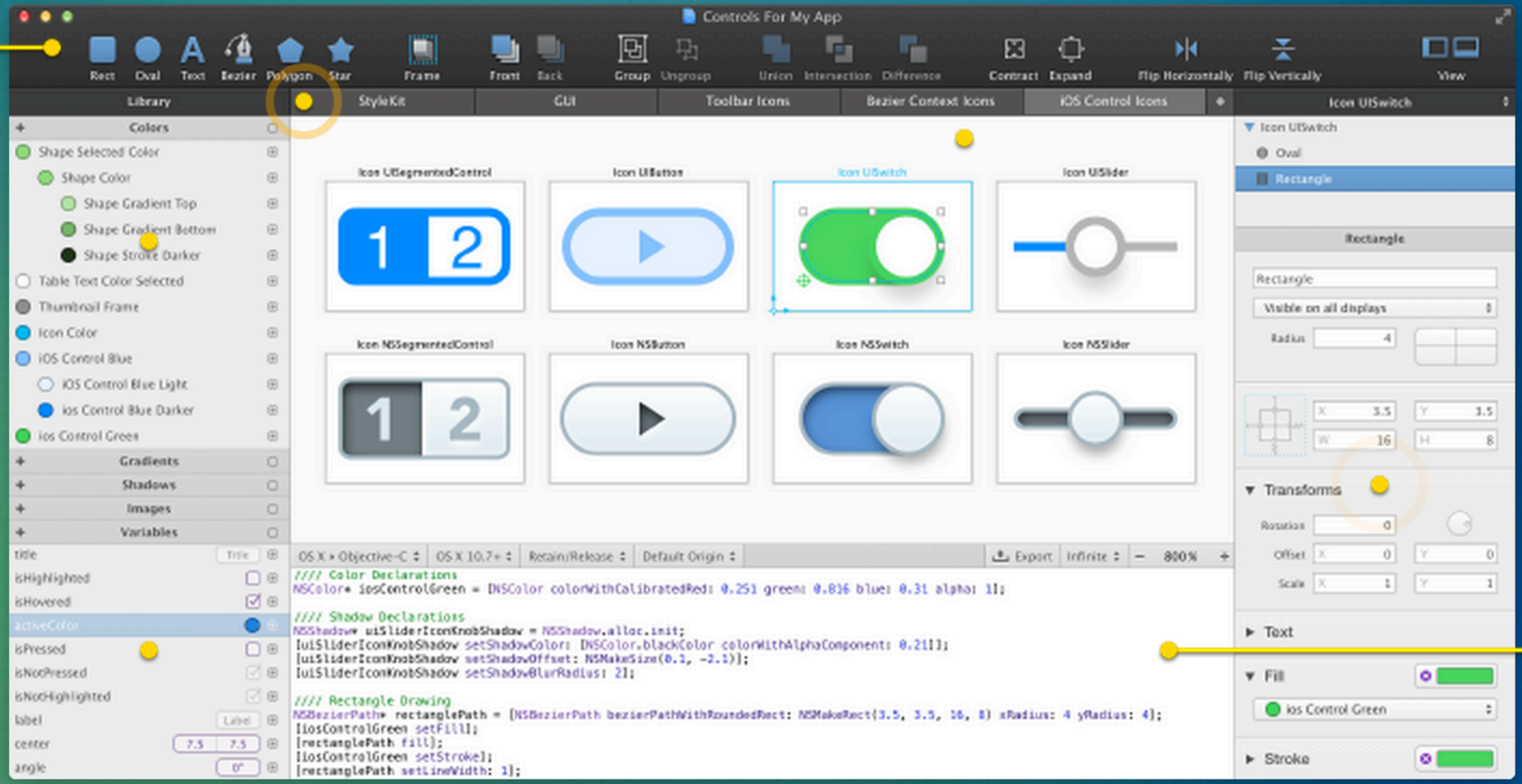
1. Subclass UIView
    - Eller UIControl om du lager interaktive komponenter
2. Override `drawRect:` for å tegne viewet (ved behov)
3. Implementerer eventhåndtering (ved behov)

The View Drawing Cycle

- Draw-kode (`drawRect:`) kalles normalt en gang og caches
  - Ikke kall `drawRect:` på egenhånd
- Ved endringer: bruk `setNeedsDisplay` som vil kalle `drawRect:` ved neste anledning

Turn drawings into code with PaintCode 2

Drawing Tools
Use built-in vector drawing tools to design controls, icons and other graphic elements.



Code Generation
PaintCode instantly generates Objective-C, Swift or C# code from your drawings.

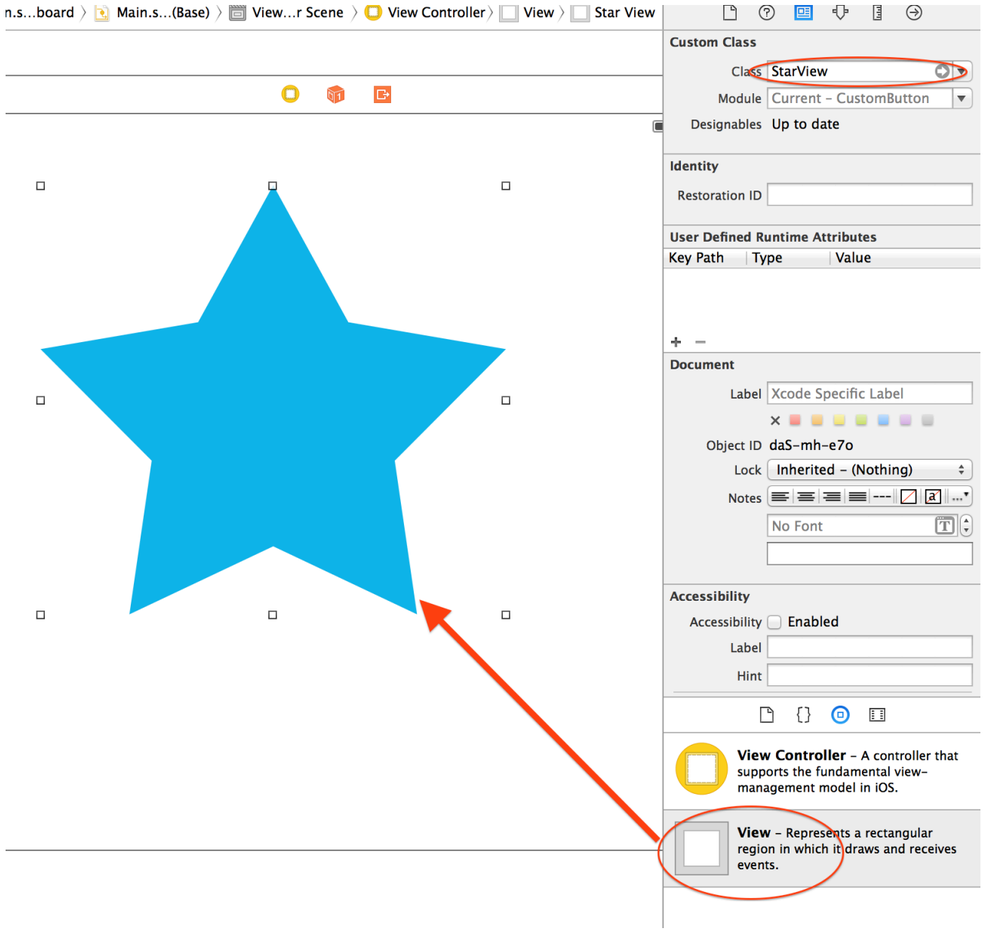
Eksempel

```
//// Color Declarations
let bgColor = UIColor(red: 0.078, green: 0.705, blue: 0.912, alpha: 1.000)

//// Star Drawing
var starPath = UIBezierPath()
starPath.moveToPoint(CGPointMake(frame.minX + 0.50000 * frame.width, frame.minY + 0.00000 * frame.height))
starPath.addLineToPoint(CGPointMake(frame.minX + 0.66095 * frame.width, frame.minY + 0.31840 * frame.height))
starPath.addLineToPoint(CGPointMake(frame.minX + 0.99872 * frame.width, frame.minY + 0.38093 * frame.height))
starPath.addLineToPoint(CGPointMake(frame.minX + 0.76042 * frame.width, frame.minY + 0.64024 * frame.height))
starPath.addLineToPoint(CGPointMake(frame.minX + 0.80823 * frame.width, frame.minY + 0.99728 * frame.height))
starPath.addLineToPoint(CGPointMake(frame.minX + 0.50000 * frame.width, frame.minY + 0.83915 * frame.height))
starPath.addLineToPoint(CGPointMake(frame.minX + 0.19177 * frame.width, frame.minY + 0.99728 * frame.height))
starPath.addLineToPoint(CGPointMake(frame.minX + 0.23958 * frame.width, frame.minY + 0.64024 * frame.height))
starPath.addLineToPoint(CGPointMake(frame.minX + 0.00128 * frame.width, frame.minY + 0.38093 * frame.height))
starPath.addLineToPoint(CGPointMake(frame.minX + 0.33905 * frame.width, frame.minY + 0.31840 * frame.height))
starPath.closePath()
bgColor.setFill()
starPath.fill()
```

Lag custom view ut av tegningen

```
@IBDesignable // <-- Gir preview i IB
class StarView: UIView {

    override func drawRect(rect: CGRect)
    {
      // tegnekode her
    }
}
```

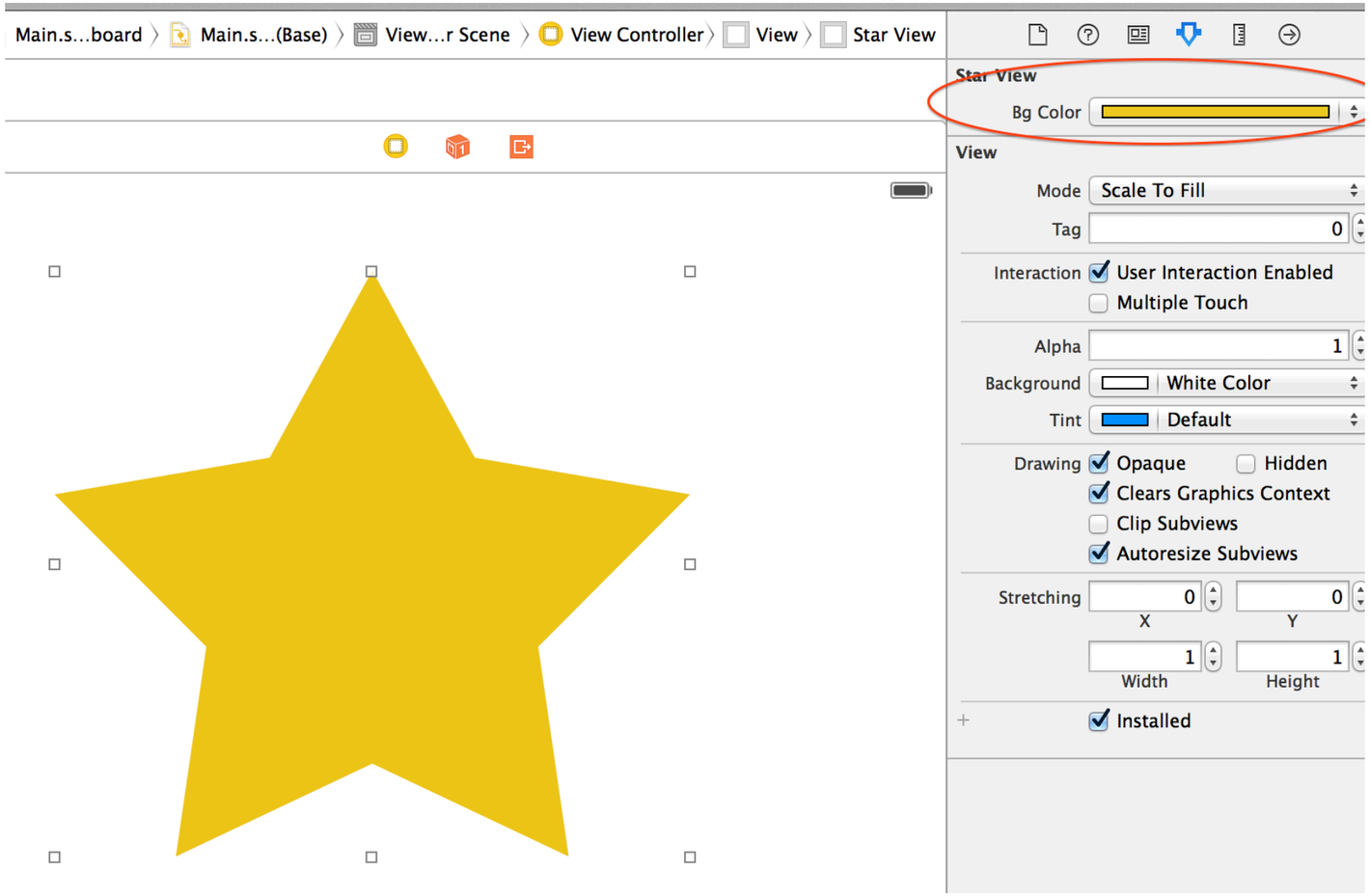
Bruk custom view i Interface builder



Custom attributter via IB

```
@IBDesignable
class StarView: UIView {

  // @IBInspectable blir tilgjengelig i IB sitt gui!
  @IBInspectable var bgColor:UIColor = UIColor(red: 0.078, green: 0.705,
      blue: 0.912, alpha: 1.000)
  //...

}
```

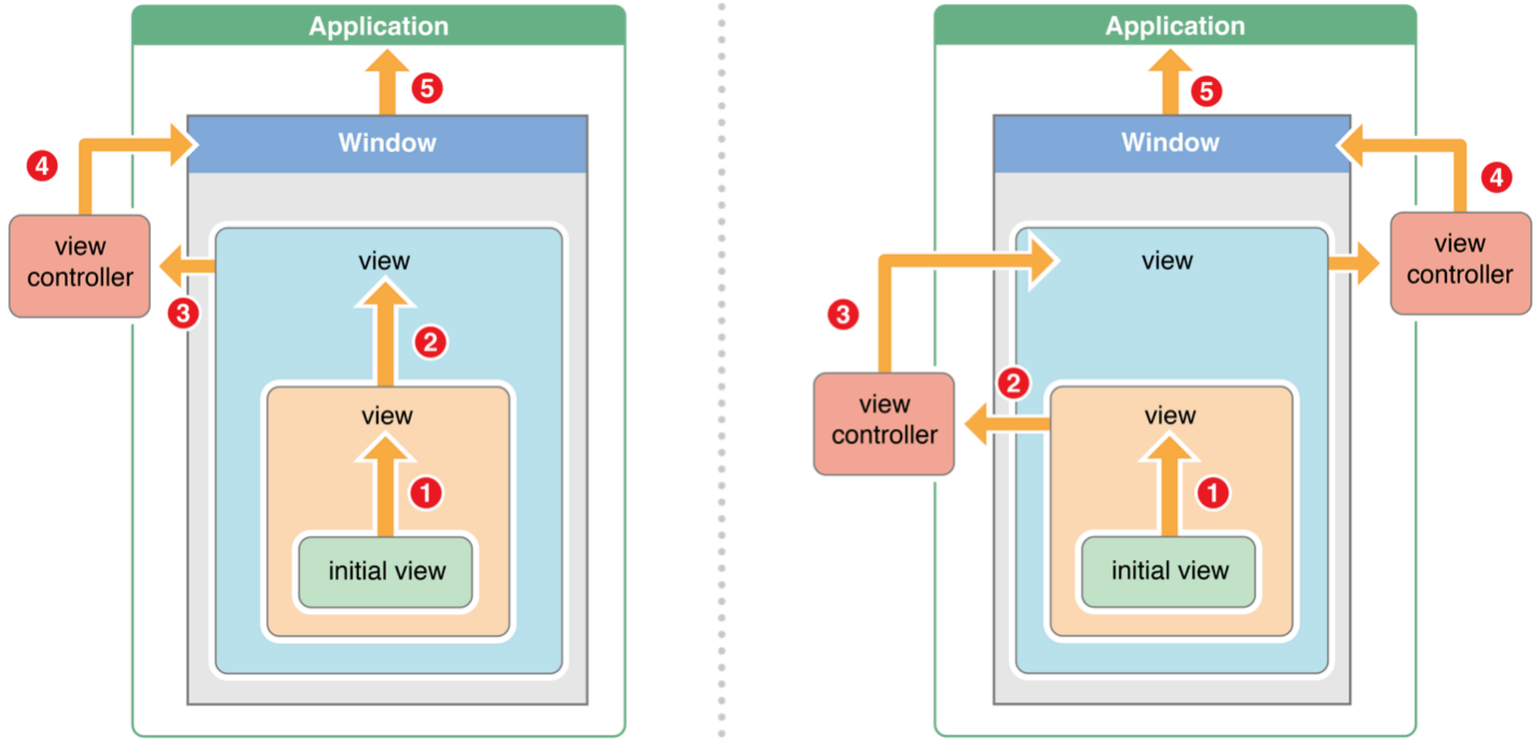
Custom attributter via IB



Eventhåndtering

Responder chain

Figure 2-2 The responder chain on iOS



Noen måter å håndtere eventer

- På egenhånd i UIView / UIViewController:
  - `touchesBegan:withEvent:`
  - `touchesMoved:withEvent:`
  - `touchesEnded:withEvent:`
- Lage view som subklasser UIControl
- Legge gesture recognizers på views

touchesEnded:withEvent:

```
@IBDesignable
class StarView: UIView {

  override func drawRect(frame: CGRect) { /* ... */ }

  override func touchesEnded(touches: NSSet, withEvent event: UIEvent) {
    let touch = touches.anyObject() as UITouch
    let point = touch.locationInView(self)
    println("Touch location: \(point)")
  }
}
```

Subclass UIControl

Når det du lager trenger Target-Action patternet og logisk sett er en UI-kontroll

```
class StarView: UIControl {
   //...
}
```

Subclass UIControl

```
// Arver fra UIControl:
func addTarget(target: AnyObject?, action: Selector,
  forControlEvents controlEvents: UIControlEvents)
// ... og hvis du vil trigge noe custom fra komponenten din:
func sendActionsForControlEvents(controlEvents: UIControlEvents)

// Som ViewController kan lytte på med:
starButton.addTarget(self, action: "tappedStar:", forControlEvents: .TouchUpInside)

func tappedStar(sender: AnyObject) {
  println("Tapped!")
}

// ... eller drag'n'drop @IBAction som vanlig
```

Hit testing

View er rektangulære:

Trykke på de hvite områdene på stjerna trigger tap. Kan løses med custom hit testing.

Hit testing

```
@IBDesignable
class StarView: UIControl {

    var starPath = UIBezierPath()

    override func drawRect(frame: CGRect)
    {
        //// Star Drawing
    }

    override func pointInside(point: CGPoint, withEvent event: UIEvent?) -> Bool {
        return starPath.containsPoint(point)
    }
}
```

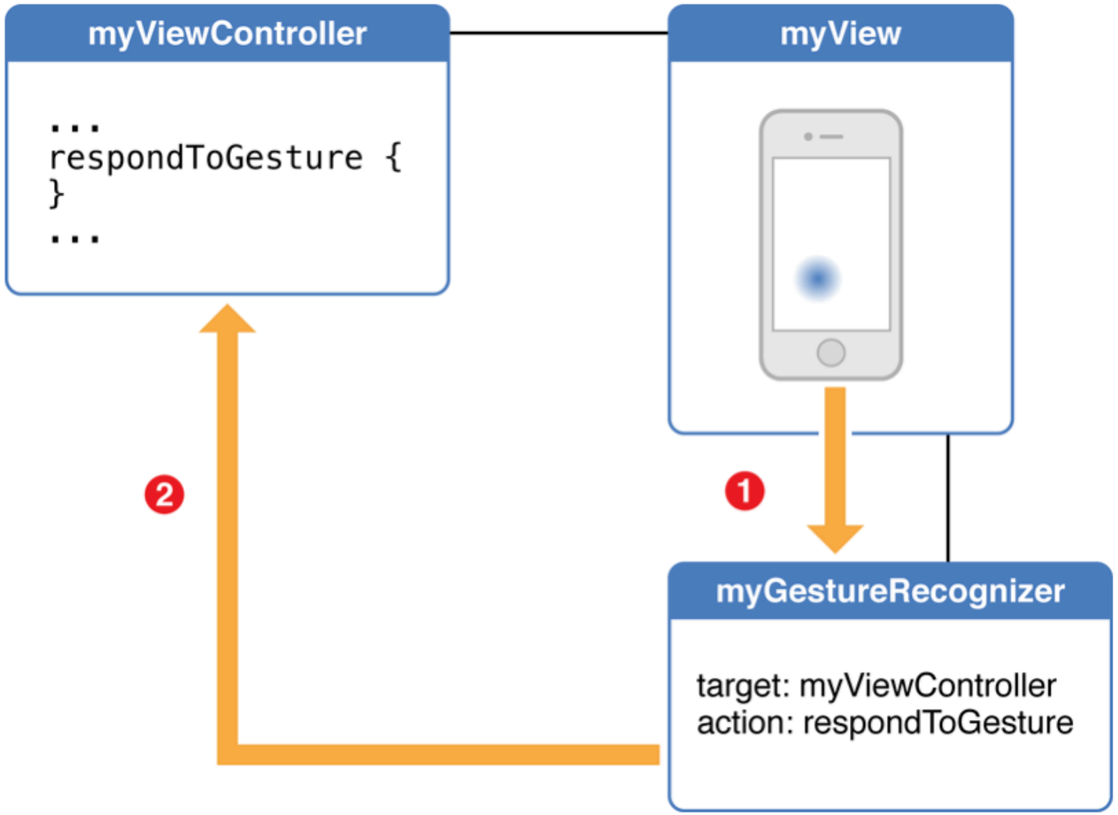
Gesture recognizers

Gesture recognizers

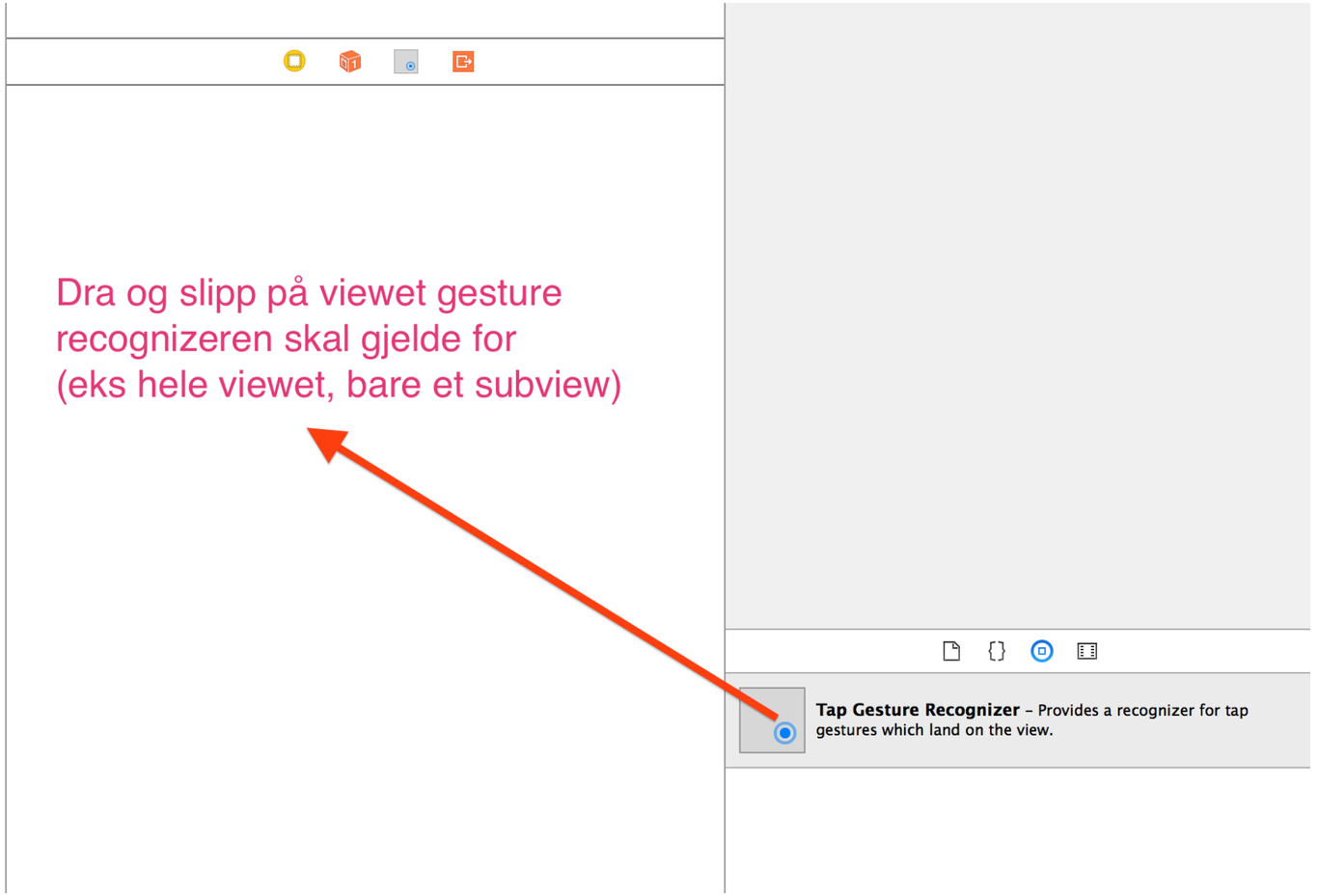
- Kan forenkler eventhåndtering ved å gjenkjenne vanlige gestures:
  - UITapGestureRecognizer
  - UIPinchGestureRecognizer
  - UIPanGestureRecognizer
  - UISwipeGestureRecognizer
  - UIRotationGestureRecognizer

Gesture recognizers

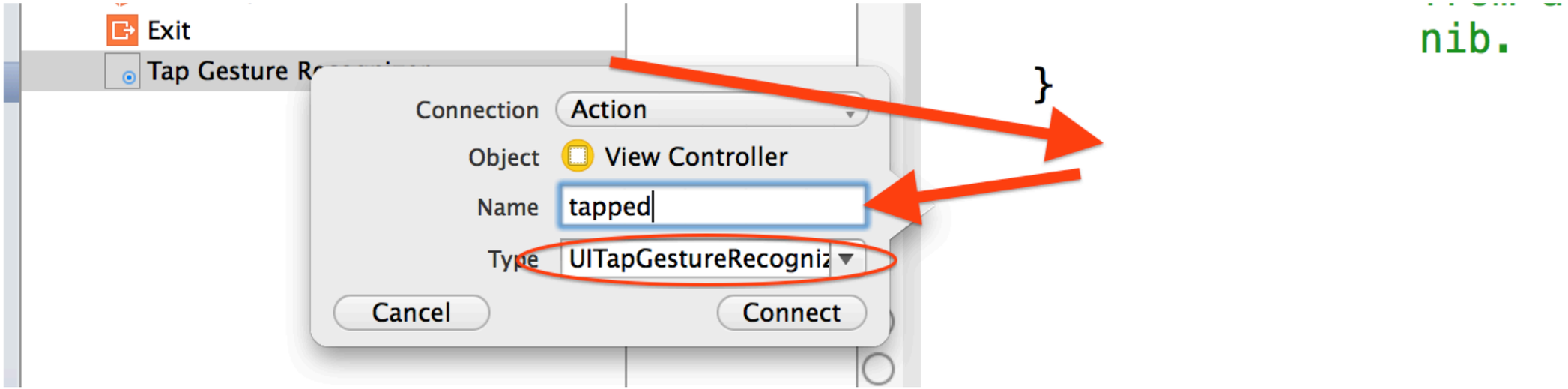
Figure 1–1 A gesture recognizer attached to a view



1. Legg GR på et (sub)view



2. Knytt action til viewcontroller



3. Hent ut info fra gesture

```
// I controller. Tegn firkanter der brukeren tappet
@IBAction func tapped(sender: UITapGestureRecognizer) {
    let point = sender.locationInView(self.view)
    let v = UIView(frame: CGRectMake(0, 0, 20, 20))
    v.center = point
    v.backgroundColor = .redColor()
    self.view.addSubview(v)
}
```

Animasjoner

Animasjoner

- Enklere animasjoner med UIKit via UIView
- Mer kontroll via Core Animation (Core Animation Programming Guide)

Viewanimasjon med blokk

```
// self.image.alpha = 1
// Vil bli en fade-out animasjon på 1. sekund:
UIView.animateWithDuration(1, animations: { () -> Void in
    self.image.alpha = 0
})
```

Nøstede animasjoner

```
// Fade ut med custom options, deretter inn igjen
UIView.animateWithDuration(0.2, delay: 0, options:
  UIViewAnimationOptions.BeginFromCurrentState | UIViewAnimationOptions.CurveEaseIn,
animations: { () -> Void in
    self.image.alpha = 0
}) { (finished) -> Void in
    UIView.animateWithDuration(1, animations: { () -> Void in
        self.image.alpha = 1
    })
}
```

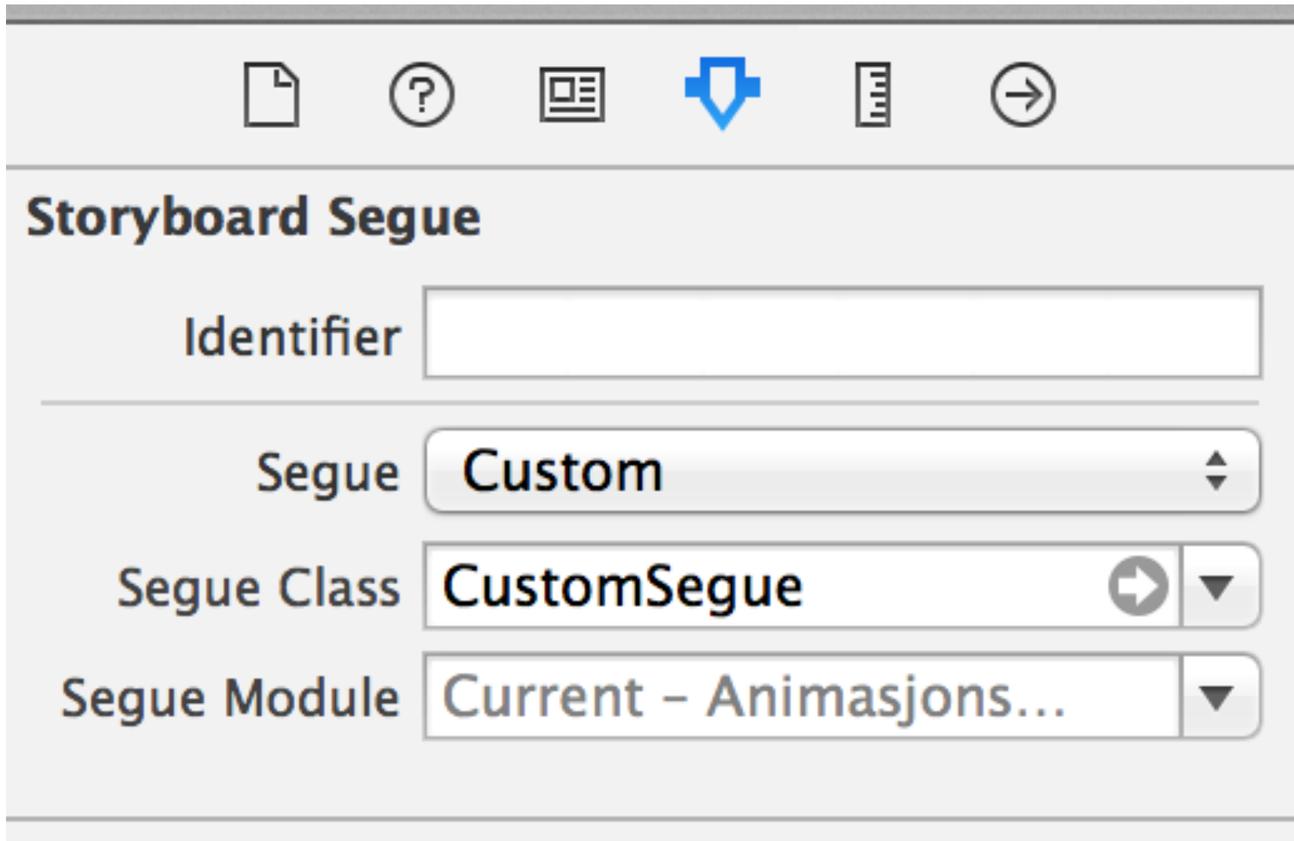
Animer hele views

Ved større endringer innenfor samme view:

```
UIView.transitionWithView(self.view, duration: 1, options:
  UIViewAnimationOptions.TransitionCrossDissolve, animations: { () -> Void in
  // Legg til/fjern/skul/vis subviews
}, completion: nil)
```

Animer custom overgang mellom view controllere (med segue)

1. Opprett en ny klasse som arver fra `UIStoryboardSegue`
2. Lag segue som vanlig mellom VC i storyboard, men velg "Custom" som type
3. Sett segue class:



```
class CustomSegue: UIStoryboardSegue {
  override func perform() {
    let source = self.sourceViewController as UIViewController
    let destination = self.destinationViewController as UIViewController

    // 1. Legg destination viewet inn i source viewet
    source.view.addSubview(destination.view)

    // 2. <sett opp starttilstand for views>

    UIView.animateWithDuration(1, animations: { () -> Void in
        // 3. <sett opp sluttilstand for views>
        }) { (finished) -> Void in
        // 4. <presenter destinationcontroller når animasjonen er ferdig>
    }
  }
}
```

```
class CustomSegue: UIStoryboardSegue {
  override func perform() {
    let source = self.sourceViewController as UIViewController
    let destination = self.destinationViewController as UIViewController

    source.view.addSubview(destination.view)

    destination.view.alpha = 0
    destination.view.transform = CGAffineTransformMakeScale(0.05, 0.05)

    UIView.animateWithDuration(1, animations: { () -> Void in
        destination.view.alpha = 1
        destination.view.transform = CGAffineTransformMakeScale(1, 1)
        }) { (finished) -> Void in
        //source.presentViewController(destination, animated: false, completion: nil)
        source.navigationController!.pushViewController(destination, animated: false)
    }
  }
}
```

Oppgaver

Se Øvingsoppgavene

https://github.com/hinderberg/ios-swift-kurs