Advanced User Interfaces

Dr.-Ing. Thomas Springer
M.Sc. Martin Weißbach
• Size Classes
• Custom User Interface Widgets
Size Classes
• Display Sizes
  • not the amount of Pixels is crucial, but
  • the number of magnitude the Display sizes differ
  • this magnitude is called ‘Size Class’
## Size Class Matrix

### Horizontal (Device Width)

<table>
<thead>
<tr>
<th>Vertical (Device Height)</th>
<th>Compact</th>
<th>Regular</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compact</td>
<td>* iPhone — Landscape</td>
<td>* iPhone * Plus — Landscape</td>
</tr>
<tr>
<td>Regular</td>
<td>* iPhone — Portrait</td>
<td>* iPad (Portrait + Landscape)</td>
</tr>
</tbody>
</table>

* … >= 6
## Size Classes Matrix

### Horizontal (Device Width)

<table>
<thead>
<tr>
<th>Vertical (Device Height)</th>
<th>Compact</th>
<th>Any</th>
<th>Regular</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compact</td>
<td>iPhone — Landscape</td>
<td>Compact Height Any Width</td>
<td>iPhone * Plus — Landscape</td>
</tr>
<tr>
<td>Any</td>
<td>Any Height Compact Width</td>
<td>Default</td>
<td>Any Height Regular Width</td>
</tr>
<tr>
<td>Regular</td>
<td>iPhone — Portrait</td>
<td>Regular Height Any Width</td>
<td>iPad (Portrait + Landscape)</td>
</tr>
</tbody>
</table>

* … >= 6
Customization with Size Classes

- Constraint constant

- Fonts

- Sets of Constraints

- Subviews
  - install or deinstall them to the scene
  - different from hidden: ‘hidden’ views still participate in the constraint equations
Customization with Size Classes

- Customizations applied to specific Size Classes are only valid within this Size Class in Interface-Builder
  - e.g. reduced bottom bar height in Compact, h:Compact is not applied to Size Class w:Compact, h:Regular
“Traits specify the size class, display scale, and idiom for a particular object.” [1]

- Size Class
  - Unspecified, Compact, Regular
- Display Scale
  - 0.0 (unspecified), 1.0 (non-retina), 2.0 (retina), 3.0 (iPhone 6 Plus)
- User Interface Idiom
  - Unspecified, Phone, Pad

- UITraitCollection — collection of all aforementioned traits
• Views can have different Size Classes on the same device
• `setOverrideTraitCollection(_:forChildViewController:)`

• Parent of a view can override the Size Class “inherited” from the system
  • e.g. `UISplitViewController` overrides Size Class Trait of Master to have `w:Compact`
Trait Environment

from: Apple WWDC 2014, Session 214, View Controller Advancements in iOS 8
Custom User Interfaces Widgets
Integration with Interface Builder
UIView and CALayer
• every view is backed up with a CALayer (sub-)class → “backing layer”
• handles user input
• many methods simply delegate to the layer, e.g.
  • changing the frame
  • modifying the alpha

• graphical representation
Problem → Solution

• non-standard controls and widgets not applicable in Interface and not customizable
• representation of custom controls and widgets with the help of UIView widgets
  • poorly integrations with Auto Layout
• @IBDesignable and @IBInspectable
Custom User Interfaces Widgets

@IBDesignable class CustomView: UIView {
    override class func layerClass() -> AnyClass {
        return CAShapeLayer.self
    }

    override func drawRect(rect: CGRect) {
        if let layer = self.layer as? CAShapeLayer {
            let frameCenter = CGPoint(x: self.bounds.width/2, y: self.bounds.height/2)
            let radius = (self.bounds.width > self.bounds.height) ? self.bounds.height/2.5 : self.bounds.width/2.5
            let path = UIBezierPath(arcCenter: frameCenter, radius: radius, startAngle: 0, endAngle: 360, clockwise: true)
            layer.path = path.CGPath
            layer.borderWidth = 2.0
            layer.borderColor = circleColor?.CGColor
        }
    }

    @IBInspectable var cornerRadius: CGFloat = 5.0 {
        didSet {
            layer.cornerRadius = cornerRadius
        }
    }

    @IBInspectable var circleColor: UIColor? {
        didSet {
            layer.borderColor = circleColor?.CGColor
            if let layer = layer as? CAShapeLayer {
                layer.fillColor = circleColor?.CGColor
            }
        }
    }
}
- `prepareForInterfaceBuilder()`
  - method to setup the custom view with default values, e.g. default string for text control
  - method is not called at runtime at all but only by Interface Builder at design time
  - must call super at some time to give super classes time for customization