User Interfaces

Dr.-Ing. Thomas Springer
M.Sc. Martin Weißbach
the prepared Xcode project for today’s lecture can be downloaded here:

https://www.rn.inf.tu-dresden.de/martin/iOS-Programming-17/User-Interface-Example.zip
Frames and Positioning

• Frame
  • describes the view’s position and size in its superview’s coordinate system

• Bounds
  • describes the view’s position and size in its own coordinate system
  • default origin is (0,0)
  • width and height is set to the frame’s width and height by default
  • used to display different areas of a view
Frames and Positioning

(\theta, \theta)

(120, 225)

Button

Frame
Button’s frame origin:

X: 120

Y: 225
Button’s frame origin is still:

X: 120
Y: 225

But the superview’s coordinate system changed
Auto Layout?
Auto Layout? What’s that?

- define how your interface is laid out by defining constraints
- constraints are mathematical descriptions of elements or between different elements
- constraints can be set up on single elements or between elements
- used to design dynamic user interfaces
  - different screen sizes
  - localization
  - orientation changes
Auto Layout? Why?

• Relational
  • user interfaces are described relatively, what can be codified with Auto Layout

• Dynamic
  • size automatically adjusted when localization changes
  • automatically adjusts to different screen sizes, e.g. 4”, 4.7” or 5.5”

• Expressive
  • relations between views instead of hard-coded coordinate positions
• views position described by relative constraints instead of a fixed origin of its frame
  • center horizontally
  • center vertically
  • button has a fixed width

• at runtime the absolute position of the view in the scene will be calculated
• automatic readjustment of the position on orientation changes
• views position described by relative constraints instead of a fixed origin of its frame
  • center horizontally
  • center vertically
  • button has a fixed width
• at runtime the absolute position of the view in the scene will be calculated
• automatic readjustment of the position on orientation changes

Why is no height explicitly specified?
Intrinsic Content Size

• Some UI widgets have an inherent width and height
  • Labels, Buttons, Textfields etc.

• this intrinsic content size mostly matches the size of the text displayed in the widget, e.g. the line height of the “Button” text of the button
  
  the example buttons design width is greater than it’s intrinsic content width
  
  therefor the width needs to be specified explicitly to resolve ambiguities
Safe Area: area of the view that is not obstructed by bars
• Most work with Auto Layout is done in Interface Builder

• alike steps to setup constraints in Storyboard and nib-files (mostly)
  • Storyboard sets up implicit constraints at runtime when unspecified at design time
• select the button
• click
• hold ctrl and move the mouse to the left
• release the mouse
• select “Center Vertically in Container”

• at runtime the button will be arranged so that the padding to the top and bottom border of the containing view is equal
Auto Layout? How?

- the orange line indicates the vertical alignment

- orange lines mean, the constraints are ambiguous
  - too few constraints in this case

- blue lines indicate enough information to resolve the specified constraints at runtime
**Intermediary States**

- **Orange Lines**
- Ambiguous Frames: Not enough information
- Conflicting Constraints: Too much information
- Misplaced Views: mismatched position or size

- e.g. the button was moved from its original position (constraints already set up) to a new location
• select the button
• click, ctrl, drag the mouse pointer below or above the button
• release the mouse and select “Center Horizontally in Container”

• at runtime the button will be arranged so that the padding to the right and left border of the containing view is equal
• view is still ambiguous

• add another constraint, but this time release the mouse over the button itself
• select “width” to add a constraint that fixes the button’s width to the value it was designed with

• all ambiguities are resolved
• blue lines indicate a working system of constraints

• the button will reside centered in the view at runtime
Align

• offers several options to align a view without using the previously shown drag’n’drop mechanism

• click “Add Constraints” to save changes when you’re done
Pin

- select at least one view
- offers several options to pin a view to another view
• Resolve Auto Layout Issues
• apply to update frames or constraints
• upper half only available for the currently selected view
• bottom half applies to all views in the scene
Additional Tooling

- Resizing Behavior
- specifies how constraints in the scene should be handled, when a view is resized
Auto Layout – WWDC Session Videos

- WWDC 2017
- (201) What’s New in Cocoa Touch

- WWDC 2013
- (406) Taking Control of Auto Layout in Xcode 5

- WWDC 2012
- (232) Auto Layout by Example
- (228) Best Practices for Mastering Auto Layout

- WWDC 2011
- (103) Cocoa Autolayout
• introduced with iOS 9
• ease relative grouping of views to each other
  • reduce placeholder views number of necessary constraints
• make user interfaces better maintainable
UIStackView – Vertically
UIStackView – Horizontally
• properties apply two horizontal and vertical dimension of a view
• the view must have an intrinsic content size for those properties to apply
Content Hugging: content does not want to grow
Content Hugging: content does not want to grow
Compression Resistance: content does not want to shrink

Superview

Higher Compression Resistance enabled for Textfield

Lower Compress...
Compression Resistance: content does not want to shrink

Superview

750

Higher Compression Resistance enabled for Textfield

749

Lower Compression Resistance enabled for Textfield

Superview

750

Higher Compression...

751

Lower Compression Resistance enabled for Textfield