Stanford CS193p

Developing Applications for iOS

Winter 2017

Today

- Autolayout
  - Review
  - Size Classes
  - Demos

Autolayout

- You've seen a lot of Autolayout already

   Using the dashed blue lines to try to tell Xcode what you intend

   Reset to Suggested Constraints (if the blue lines were enough to unambiguously set constraints)

   Size Inspector (look at (and edit!) the details of the constraints on the selected view)

   Clicking on a constraint to select it then bring up Attributes Inspector (to edit its details)

- What else?

   Ctrl-dragging can be done between views, not just to the edges

   There are "pin" and "arrange" menus in the lower right corner of the storyboard

   Document Outline is the place to go to resolve conflicting constraints

- Mastering Autolayout requires experience

   You just have to do it to learn it

- Autolayout can be done from code too

   Though you're probably better off doing it in the storyboard wherever possible

Autolayout

◉ What about rotation?

    Sometimes rotating changes the geometry so drastically that autolayout is not enough

    You actually need to reposition the views to make them fit properly

◉ Calculator

    For example, what if we had 20 buttons in a Calculator?

    It might be better in Landscape to have the buttons 5 across and 4 down

    Versus in Portrait have them 4 across and 5 down

◉ View Controllers might want this in other situations too

    For example, your MVC is the master of a side-by-side split view

    In that case, you'd want to draw just like a Portrait iPhone does

◉ The solution? Size Classes

    Your View Controller always exists in a certain "size class" environment for width and height

    Currently this is either Compact or Regular (i.e. not compact)

Autolayout

◉ **iPhone**
   iPhones in Portrait are Compact in width and Regular in height
   But in Landscape, most iPhones are treated as Compact in <u>both</u> dimensions

◉ **iPhone 6+ and 7+**
   The iPhone Plus in Portrait orientation is also Compact in width and Regular in height
   But in Landscape, it is Compact in height and <u>Regular</u> in width

◉ **iPad**
   Always Regular in both dimensions
   An MVC that is the master in a side-by-side split view will be Compact width, Regular height

◉ **Extensible**
   This whole concept is extensible to any "MVC's inside other MVC's" situation (not just split view)
   An MVC can find out its size class environment via this method in UIViewController ...
   `let mySizeClass: UIUserInterfaceSizeClass = self.traitCollection.horizontalSizeClass`
   The return value is an enum `.compact` or `.regular` (or `.Unspecified`).

Size Classes

	Compact Width	Regular Width
Compact Height	iPhones (non-Plus) in Landscape	iPhone Plus in Landscape
Regular Height	iPhones in Portrait or Split View Master	iPads Portrait or Landscape
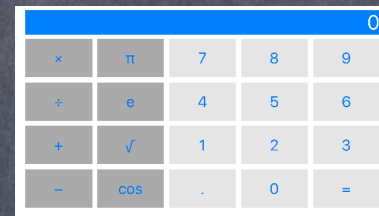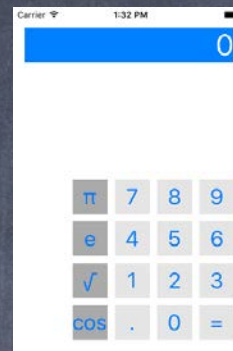
Size Classes

Demo

- Calculator

  Let's make our Calculator adjust to the size class environment it's in