Stanford CS193p

Developing Applications for iOS
Spring 2016

Today

- Autolayout
  - Review
  - Size Classes
  - Demos
- Final Project Requirements
  - What and when

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 6+
  - The iPhone 6+ in Portrait orientation is Compact in width and Regular in height
  - In Landscape, it is Compact in height and Regular in width
- iPhone
  - Other iPhones in Portrait are also Compact in width and Regular in height
  - But in Landscape, non-6+ iPhones are treated as Compact in both dimensions
- 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` …

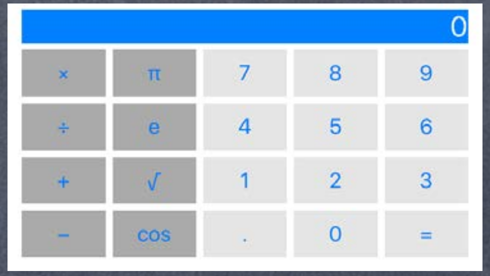
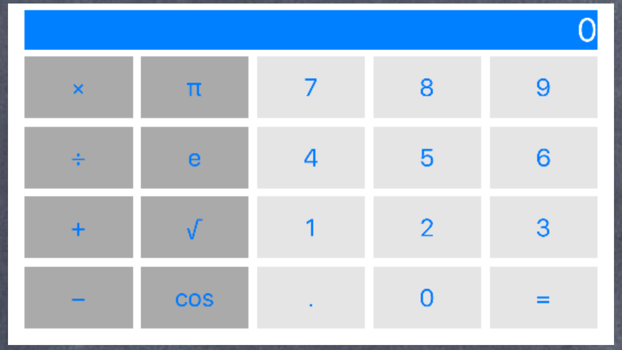
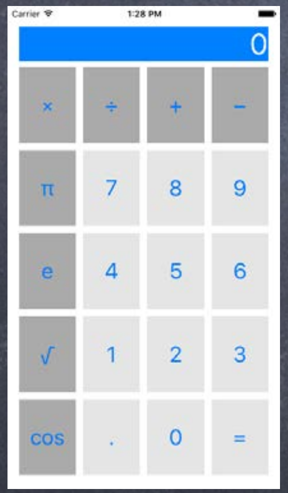
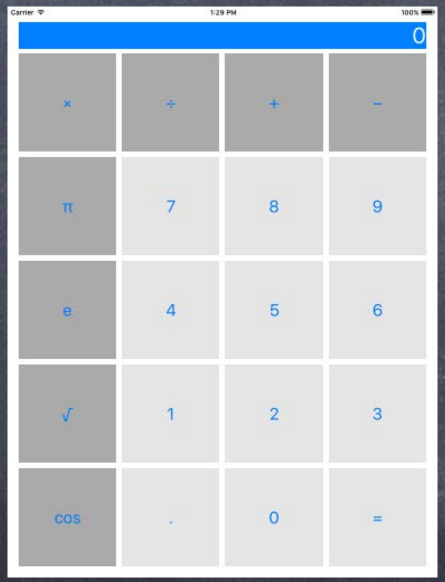
```
let mySizeClass: UIUserInterfaceSizeClass = self.traitCollection.horizontalSizeClass
```

The return value is an enum `.Compact` or `.Regular` (or `.Unspecified`).

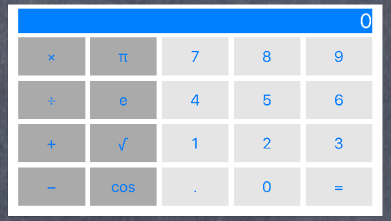
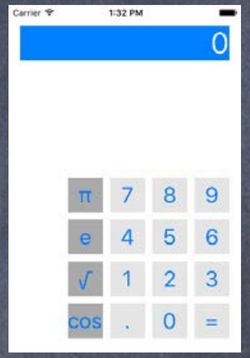
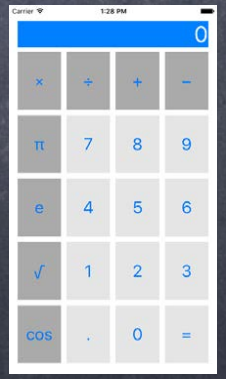
Size Classes

	Compact Width	Regular Width
Compact Height	iPhones (non-Plus) in Landscape	iPhone 6 Plus in Landscape
Regular Height	iPhones in Portrait or Split View Master	iPads Portrait or Landscape

Size Classes

	Compact Width	Regular Width
Compact Height		
Regular Height		

Size Classes

	Compact Width	Any Width	Regular Width
Compact Height			
Any Height			
Regular Height			

Demos

- Calculator

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

Final Project

- Proposal due asap!
  - And must be received no later than next Monday, May 9th.
  - Send PDF of your proposal to your CA (the one who has graded your latest assignment).
  - Proposal must say not only what you are doing, but also what parts of SDK will be featured.
  - More on the proposal in a few slides ...
- Project is due on Sunday, June 5th at midnight!
  - Use normal submission process.
  - You cannot use your "free" late days on the Final Project.
  - LATE PENALTIES ARE PROHIBITIVE (approximately one full letter grade per day late).
  - No extensions will be granted.
  - Submit whatever you have by midnight on the 5th!
  - Start (and finish) early, that way if you have an emergency at the end, it won't be a problem.
  - This is a 3 week long project (so an emergency in the last few days should have a small effect).
  - Programming is like solving crossword puzzles: you must give your subconscious time to work.
  - So don't cram 3 weeks of programming into 3 days.

Final Project

- Teams?

  Teams of a maximum of 2 are allowed only under the following circumstances:

  Both students are taking the course CR/NC.

  Both students' homework grades have been good and substantially similar through A5.

  A proposal representing 6 weeks of homework time worth of work must be explicitly approved.

  This proposal is mandatory for teams and must be received by us by Monday, May 9th.

  The sooner you submit your proposal, the sooner we'll be able to approve it.

  If you do not receive explicit approval and go ahead anyway, you risk receiving an NC.

  The proposal must clearly delineate who is going to work on what.

  Submitted code must also clearly delineate (in comments) who did what.

  Work must generally be divided up along class boundaries

  (i.e. an entire class should be written by one individual).

Final Project

- You'll be graded on proper use of SDK
  - Hackery will count against you. Use good object-oriented programming technique.
  - Must have at least one significant feature which was NOT taught in lecture!
  - Also, breadth is VERY important. Don't get stuck down a rathole.
  - Only need to (allowed to) show depth in one or two areas. Breadth is more important.
- Project scope is the same as about three weeks of homework
  - All students (including CR/NC) must pass both homework and final project segments separately.
- Do not just regurgitate the homework assignments
  - You've already proven to us you can do what the homework assignments ask.
  - Now show us that you really understood that stuff and use iOS in different contexts.
  - Start from scratch (don't start with one of your homework assignments).
  - Take concepts to the next level. Be creative.

Final Project

- Examples of bad projects
  - Smashtag 3.0 or "thing that looks a lot like Smashtag with the names changed"
  - Lots of MVCs with only buttons, labels and text fields and nothing else (i.e. little breadth)
  - Lots and lots of table views and little else (i.e. this project would lack breadth)
  - An animated game (no matter how cool) with no table views or buttons or labels or … (breadth)
  - A really cool mathematical problem solving app with little UI (this is an iOS course)
- Aesthetics of your user-interface matter
  - (although we do not expect professional graphic designer quality graphics)
  - Sloppy layouts will be graded down.
  - Lots of places to get graphics from on the internet.
- Must work on hardware!
  - If you do a live demo on hardware, obviously you will have satisfied this.
  - Otherwise you must bring your hardware to the final exam (or before) to demo for TA.
  - iPad or iPhone or iPod Touch okay.

Final Project

- Be careful not to get side-tracked on non-iOS-code

  Some students in the past have spent 80% of their time working on stuff that didn't demonstrate their mastery of the course material (e.g. preparing some large database or working on graphics too much, etc.)

  This is an iOS DEVELOPMENT course! Show us how well you can develop for this platform.

  Don't waste your time writing server-side code.

  It's okay to "simulate" a server-side interaction to make your code demonstrable.

  Third party libraries are allowed but will buy you little credit for them.

  Third party libraries also do NOT count as a "not covered in lecture" feature.

  For example, better to write your own "bar graph view" (with `drawRect`, etc.) than use CorePlot (even if your plot is not as snazzy-looking as CorePlot might have drawn).

  Bottom line: Only your code that uses the iOS SDK will "count".

Final Project

- Required 2 minute presentation during final exam period

  Wednesday, June 8th, at 3:30pm in this room (regularly-scheduled final exam period)
  or alternate time slot on Wednesday, June 1st at 4:30pm (last regularly-scheduled lecture).

  Slides (Keynote preferred) must be 1280x720 aspect ratio (not 1024x768 or 800x600).
  If you are presenting on the 1st, your slides must be submitted by Monday, May 30th.
  If you want to present on the 1st, all you need to do is submit your slides by the 30th.
  Otherwise slides are due at the same time as the project itself.
  Submit slides using normal submit process (but different target than final project itself).

Final Project

- Presentation Quality Matters
  - Not okay to just put up a recording of you or of your application and say nothing.
  - Being able to make a live presentation is a valuable skill.
  - Practice your presentation before you show up.
  - You only get 2 minutes (strictly enforced), so make 'em count.
- Live demo?
  - All iOS 9 devices (iPad2+, iPhone4s/5/6/6+) can mirror their screen to the projector here.
  - Live demos are perilous, as you saw all quarter :), but can be very effective!
  - You must, at worst, show screen shots of your application.
  - If you don't want to go live ...
    - Keynote/Quicktime has some tools to "animate" screen shots (better than static).
    - Video (screen capture) of your app in action can be good also.

Sample Proposal

- Section 1: What am I doing?

  I will be building a "Shakespeare Director" application.
  It will have the following features:
  - A table for choosing a Shakespearean play from a list downloaded from Folio*.
  - A custom view for laying out the blocking** of a chosen Shakespearean play.
  - A dialogue-learning mode.

  * Folio is an on-line database of all of Shakespeare's works.
  ** "blocking" is where people stand on stage.
  The custom view will be simple (only rectangles and circles with colors for stroke/fill, and text).
  Photos (from Camera or Library) can be put in rectangles in the blocking view.
  The blocking can change from line to line in the dialog (but no more often than that).
  Blocking can be stepped through, line by line, or played back in "time lapse" mode.
  The dialogue-learning mode will step through all the dialog line by line.
  Users can record the dialog for other parts (as prompts for them to learn their own part).
  iPad only.

Sample Proposal

- Section 2: What parts of iOS will it use?

  `UITableView` for choosing plays and stepping through dialog

  Custom `UITableViewCell` prototypes (for dialog, including speaker, blocking instructions)

  Custom `UIView` with `drawRect:` for scene-setting

  Camera/Photo Library for putting images in blocking rectangles

  `UITextField` in a `UIPopoverController` for text labels in the scene-setting view

  `UIPopoverController` for choosing stroke and fill color and shape in scene-setting mode

  Scroll view to zoom in/pan around in blocking view

  `AVFoundation` for record/playback of dialog

  `NSTimer` for "time lapse playback" of entire play with dialog/blocking linked

  Core Data to store the scene-setting and dialog

  - Play entity
  - Scene entity
  - BlockingElement entity
  - LineOfDialog entity

  Will support printing the blocking via AirPrint (a NOT COVERED IN LECTURE feature)

Sample Proposal

- What to notice about this sample proposal?
  - Clear description of what the application will do (section 1).
  - Clear list of the iOS features that will be used (section 2).
  - Lots of breadth (not necessarily that much depth in any one area).
  - Clearly delineates the NOT COVERED IN LECTURE feature(s).
  - Specifies platform (iPad-only sacrifices breadth, but makes sense for this project).
  - It's creative (it's not just Calculator or Smashtag recycled).