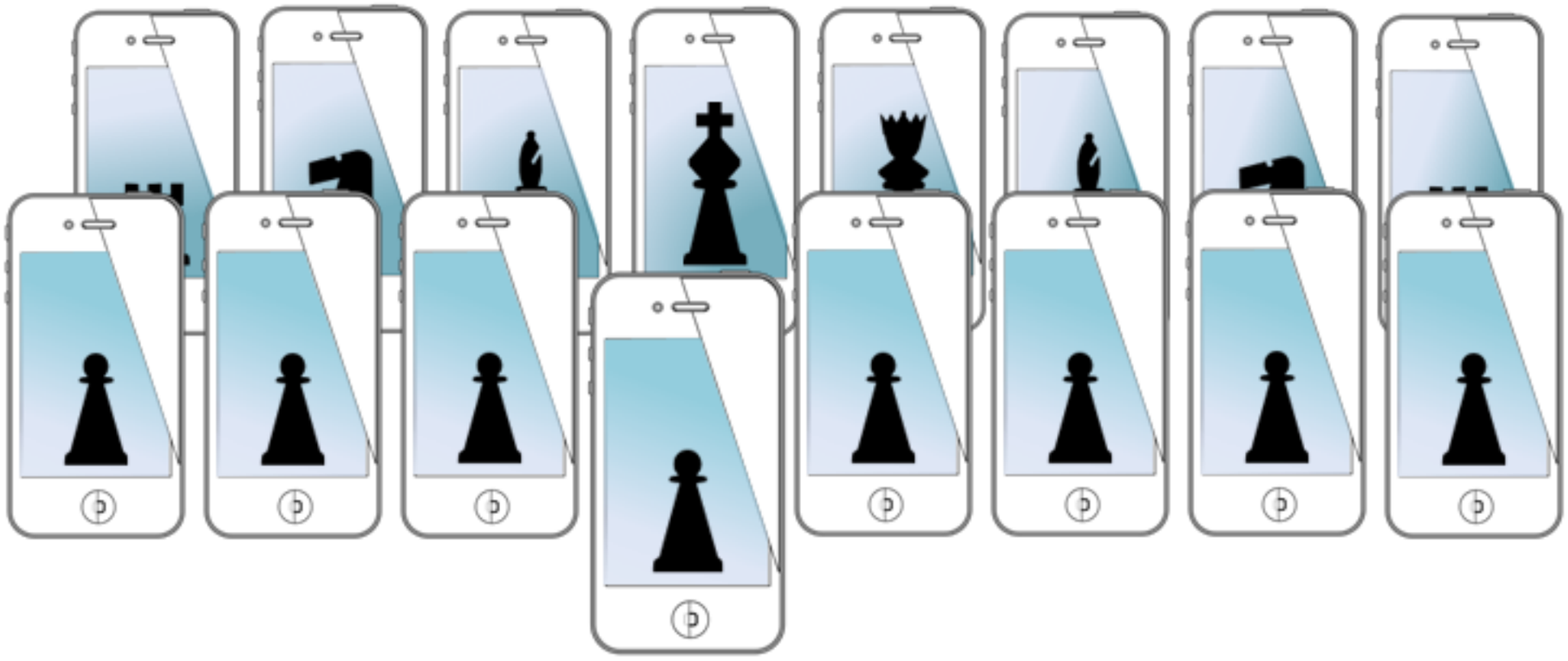
MOBILE SENSING & LEARNING



CS5323 & 7323

Mobile Sensing and Learning

course introduction

Eric C. Larson, Lyle School of Engineering,
Department of Computer Science, Southern Methodist University

agenda

- class logistics
- introductions
- what is this mobile sensing course?
  - and what this course is not…
  - course goals
  - syllabus (i.e., how to do well)
    - hardware, lab, grading, MOD
- iOS development platforms

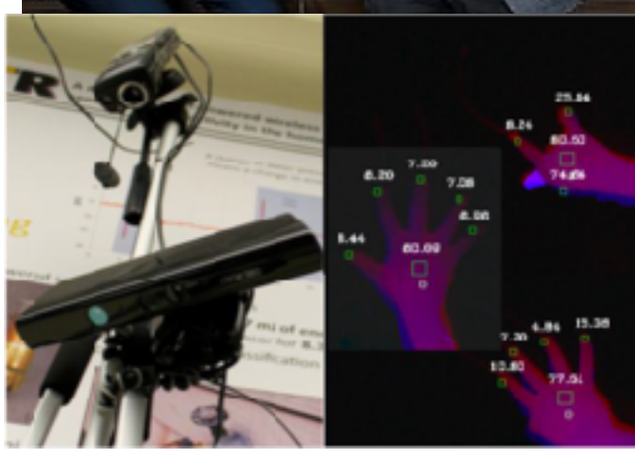
course logistics

- lecture: in class, zoom, and recorded
- **lab: this class has no lab!**
- office hours: Monday 3:30-5PM (Caruth 451, zoom)
- TA currently being on-boarded, will also have office hours
- we will use canvas/GitHub for managing the course
- and GitHub for managing code:
  - https://github.com/SMU-MSLC
- Zoom étiquette

introductions

- education
  - undergrad and masters from Oklahoma State
  - PhD from the university of Washington, Seattle
- research
  - signal, image, and video processing (mobile)
    - how can combining DSP, machine learning, and sensing make seamless computing?
  - security
    - smartphone side channels
  - mobile health
    - moving outside the clinic: how mobile sensing can help patients and doctors
  - sustainability
    - how technology can increase awareness

http://eclarson.com



SMARTPHONES

The sound of things to come?

SMU research finds new way to snoop: vibration of typing is translatable

By JORDAN WILKERSON
Staff Writer

Smartphones are like living things. With their cameras and microphones, they can see and hear. They can detect the amount of ambient lighting, the air pressure and the temperature — among a host of other aspects about the environment they're in.

Six years ago, less than half of Americans owned a smartphone. Four out of five own one now, says the Pew Research Center. There are millions of people walking around every day with a vast array of these sensors in their pockets.

And smartphones can record all of it.

This has created major concern about how easily ones privacy can be invaded by these sensor-rich devices, with partic-

See RESEARCH Page 4B

introductions (limited)

- me: Dr./Professor/ (Eric, if PhD student)
- about you:
  - name (what you go by)
  - grad/undergrad
  - department
  - **something true or false**

what is this course?

- mobile sensing
  - activity recognition **yes, via developer APIs!**
  - audio analysis **yes, custom sample access!**
  - vision analysis **yes, via multiple APIs!**
- machine learning **yes! treated as black box**
- microcontroller communication **no, archived**
- general iOS development **some but won't get you a job…**
- UI/UX, animation and graphics **not really… but a little**
- VR/AR and games **some, part of bonus lecture content**

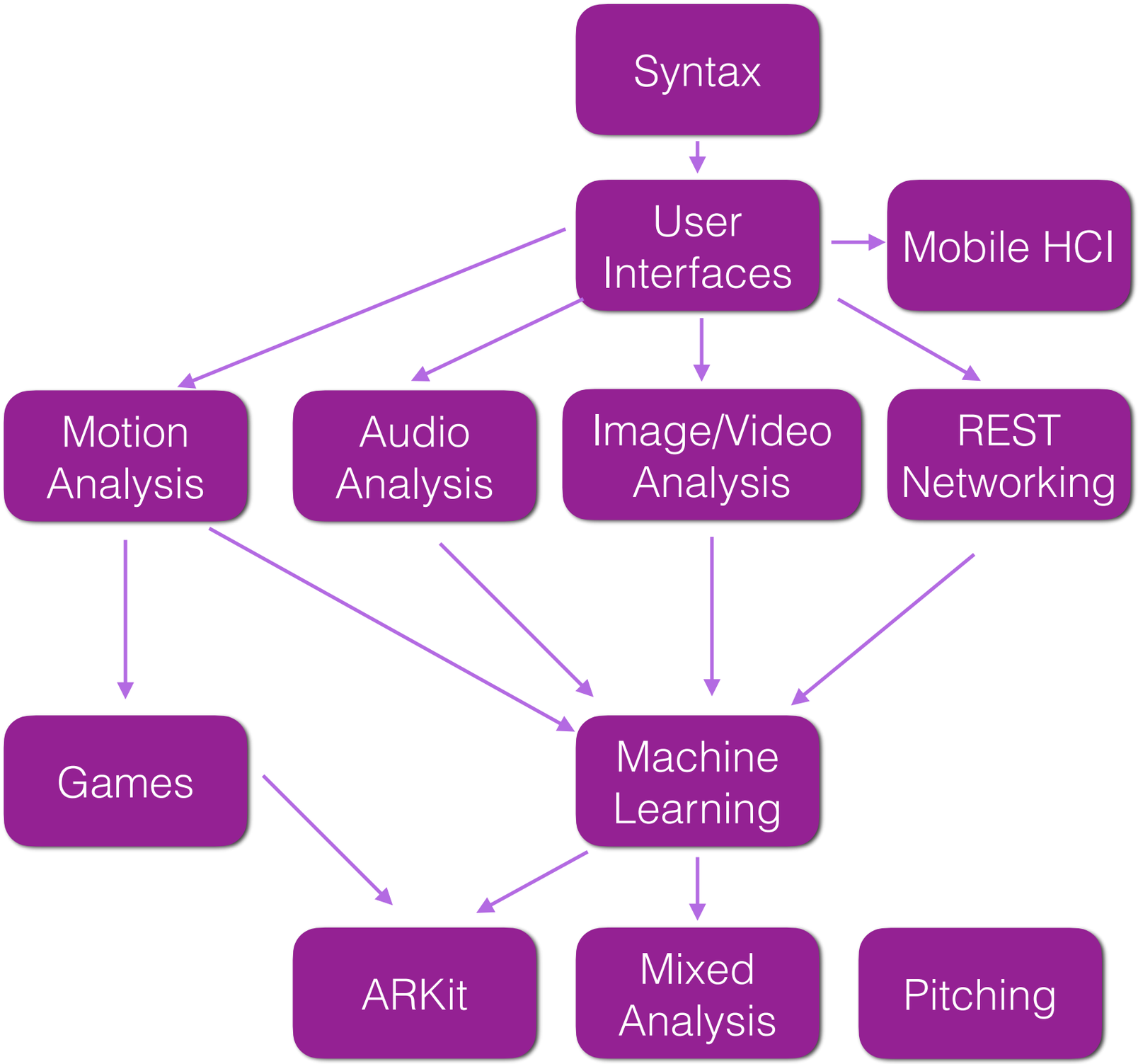
learning to learn

- for what we don't cover: take the free Stanford iOS course!
- prerequisite: model based coding
  - because you will likely learn at least one new language:
    - objective-c
    - swift

    ← mostly these, **prefer swift**
  - python ← for REST serving and ML training
    - c++
    - objective c++

    ← only for OpenCV

class overview



course goals

- exposure to iOS development, **MVCs** (not MVVM, SwiftUI)
- understand how to **use embedded sensors**
- **exposure** to **machine learning** for mobile sensors
  - use of built-in ML in iOS via coreML
- real time analysis of data streams
    - applications in health, education, security, etc.
- **present** and **pitch** applications

how to do well

- **come to class** or watch videos
- **start the app assignments early, with your team**
- iterate and test your apps
- use good coding practices, lazy instantiation, recycle classes, **use comments** (I grade comments)
- embrace generative coding, but **learn from it**

syllabus

- attendance
  - highly recommended, but you can watch video if needed
  - video of classes through Panopto (published after class)
- hardware is needed to develop apps
  - need a team formed (do this before the end of the week)
    - teams are expected to work remotely together
  - iPhones available for checkout, Xcode in library
  - preferable (required?) to use your own Mac
- Now let's head over to canvas

syllabus (via canvas)

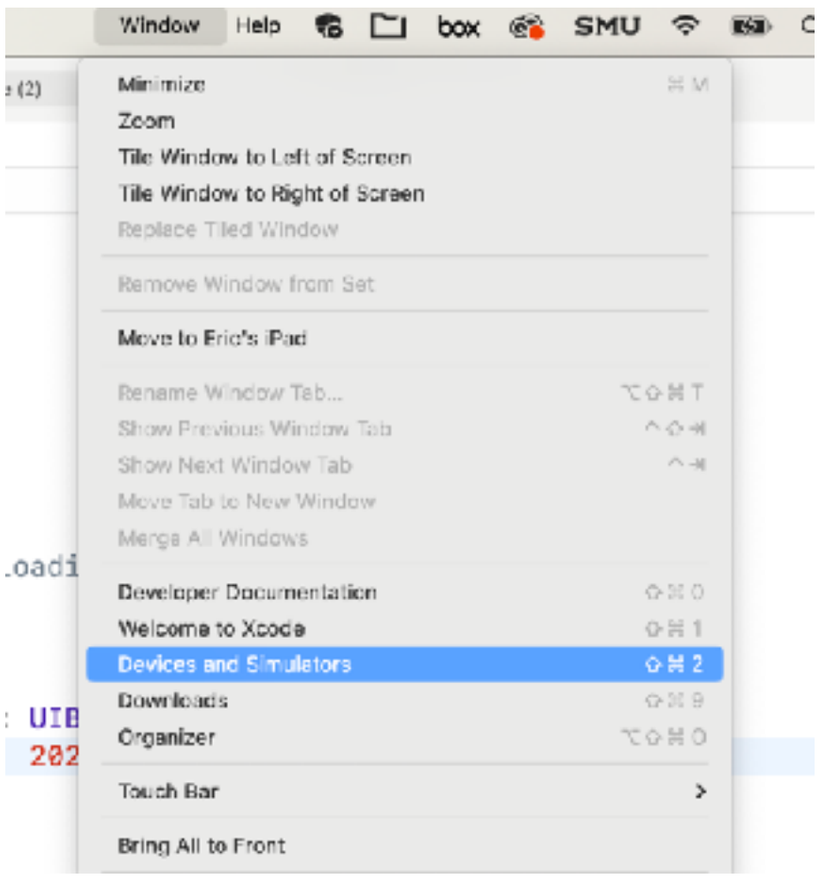
- grading
- flipped assignments
- final projects
- MOD

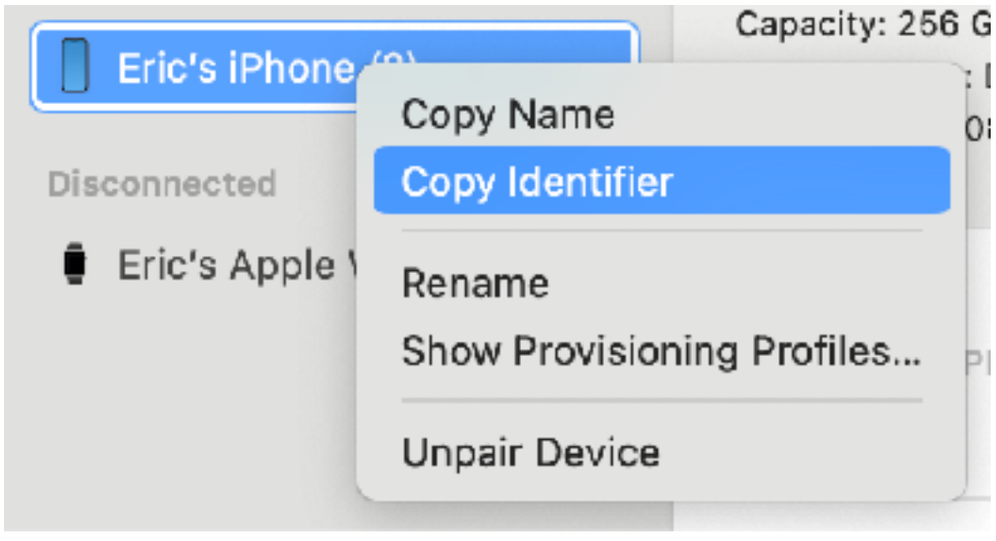
before next class

- look at canvas and GitHub repository (clone first repository)
- get a team together (groups of 1-4, no exceptions)
  - contribute **equally**, **everyone** codes, **everyone** designs
  - **pick good members** with different skills than you
  - take turns **coding**
  - **you can change teams throughout semester**
- send me information for the developer program
- all assignments are already posted for the semester and all flipped module videos

Apple Developer Program

- university developer program has been discontinued
- I will use my personal license, so send me:
  - **add user**: email that you want invite sent to (requires sign up)
  - **add device**: identifier, Xcode “Window>Devices and Simulators”

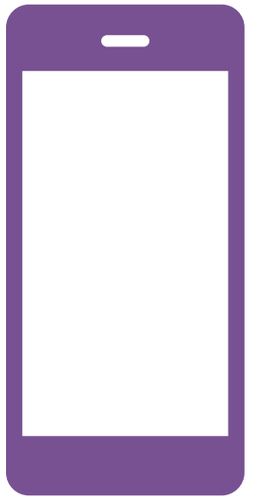




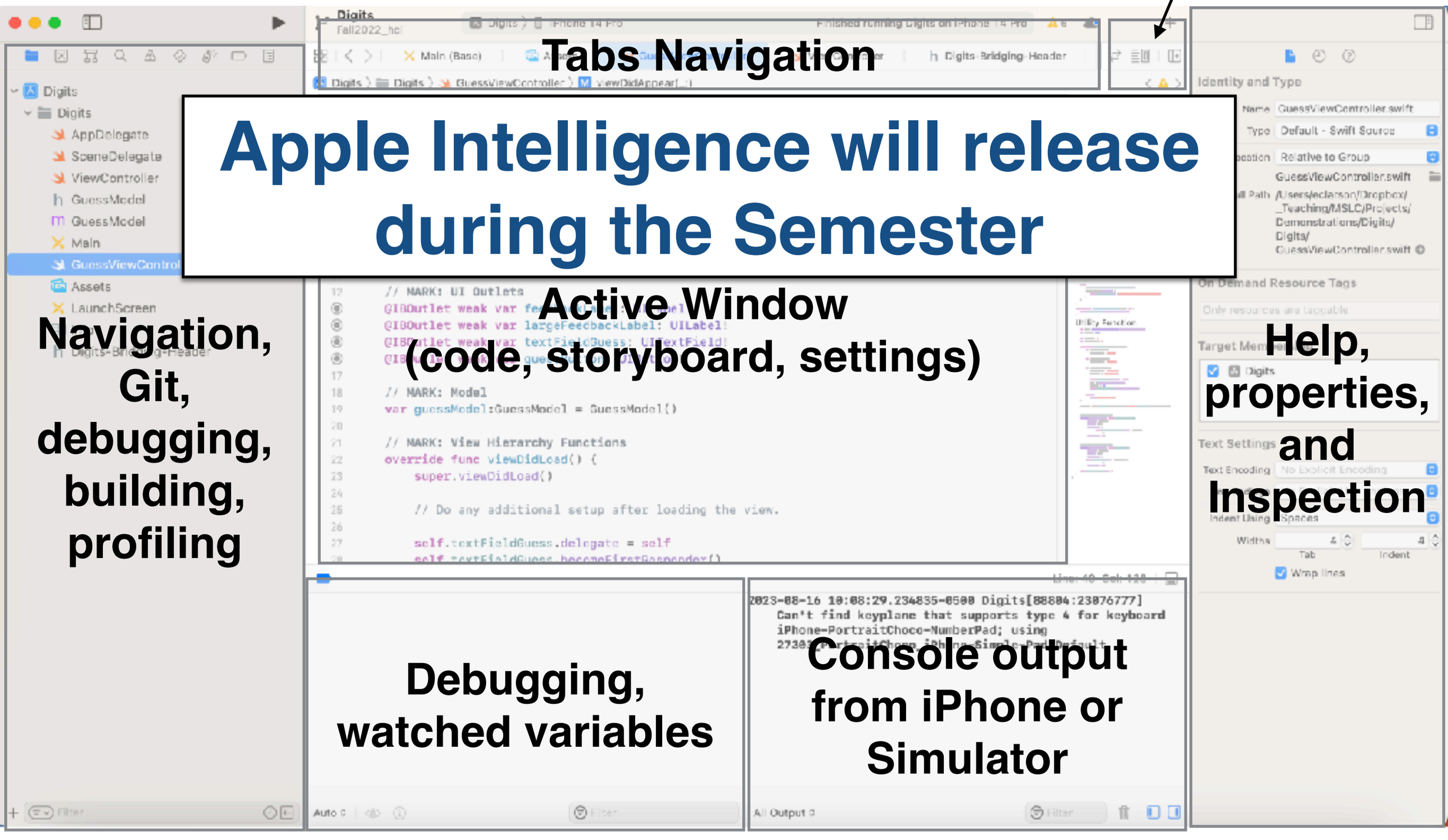
developing in iOS

developing in iOS

- **cross platform**
  - React native, flutter, MAUI, and others…
  - (-) do not always support the latest hardware capability
  - (-) using some phone sensors can be a pain (or slow)
  - (+) works on many different phones
- **native, using Apple IDE (what we will be using)**
  - (-) free to develop, but requires license to run on hardware
  - (+) packages are well supported, constantly updated
  - (+/-) limited to ONLY Apple (but also optimized for hardware)
  - (+/-) Xcode with objective-c and swift



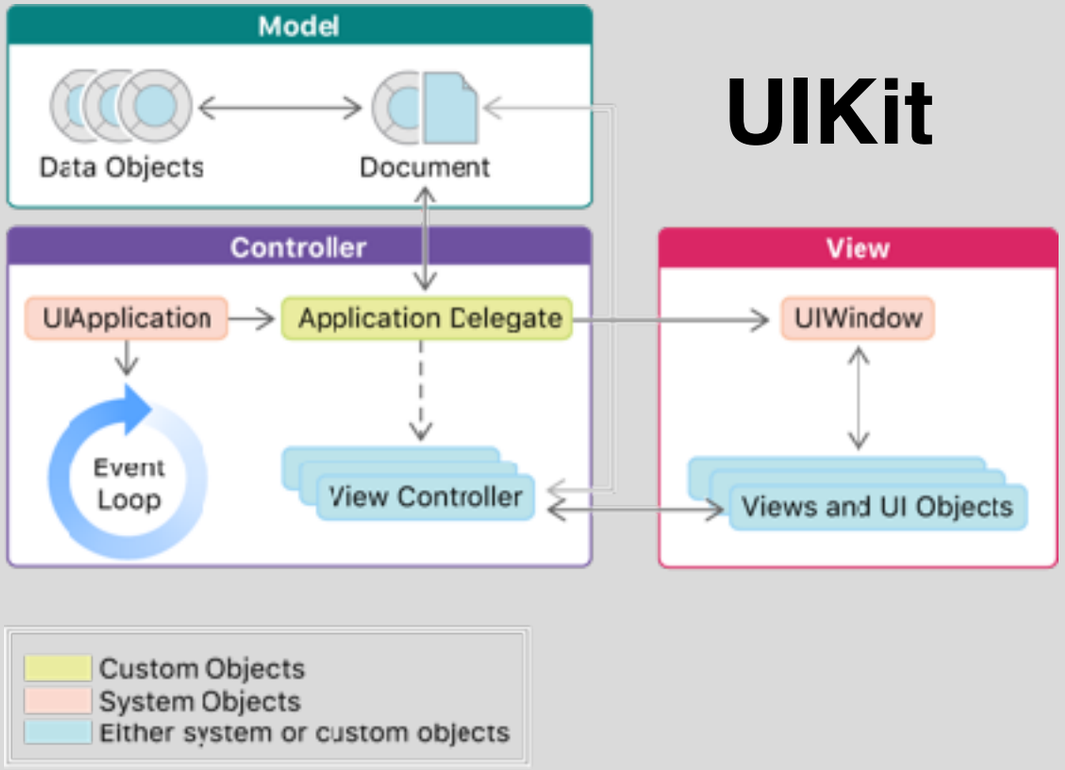
Xcode overview



Xcode

- best shown by example
- a very powerful and complex IDE
- if you update iOS, you must update Xcode
  - some Xcode updates require the newest MacOS
- support UI design through **SwiftUI** or **UIKit**
  - my exampled will use UIKit exclusively (easier to get started)
  - required use of storyboards and auto layout
  - auto layout is great for apps in one layout

UIKit versus Swift UI



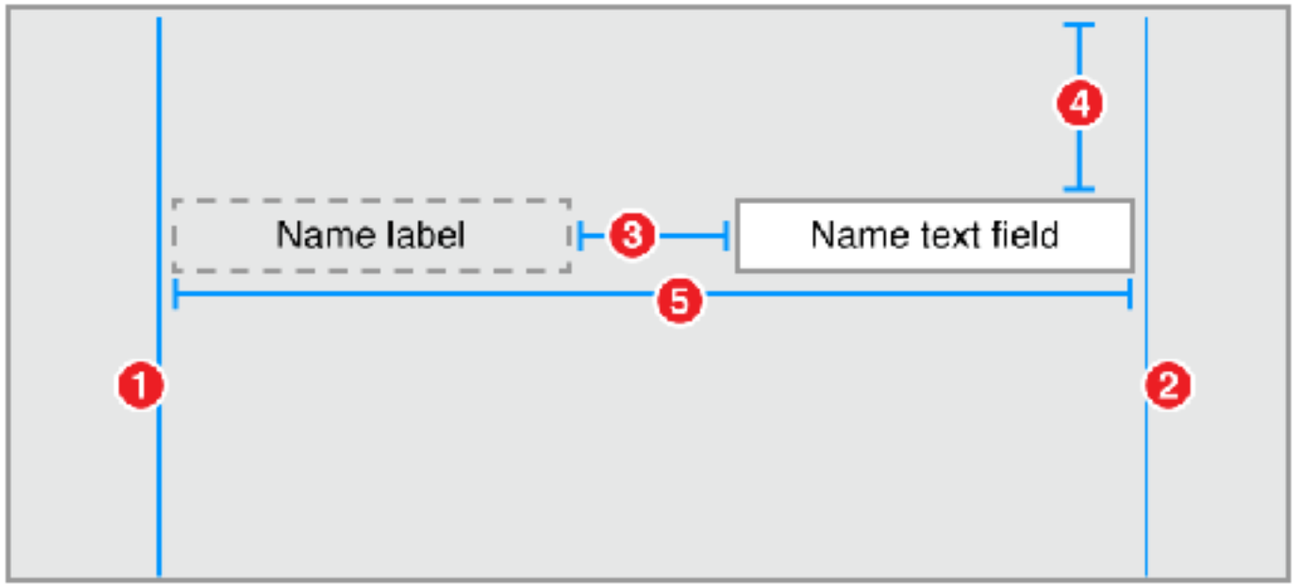
UIKit

- **UIKit** assumes develop in code and storyboard
- Requires Xcode UI to layout interfaces and link code to view
- Easy to learn and get apps running
- Harder to master for full app development

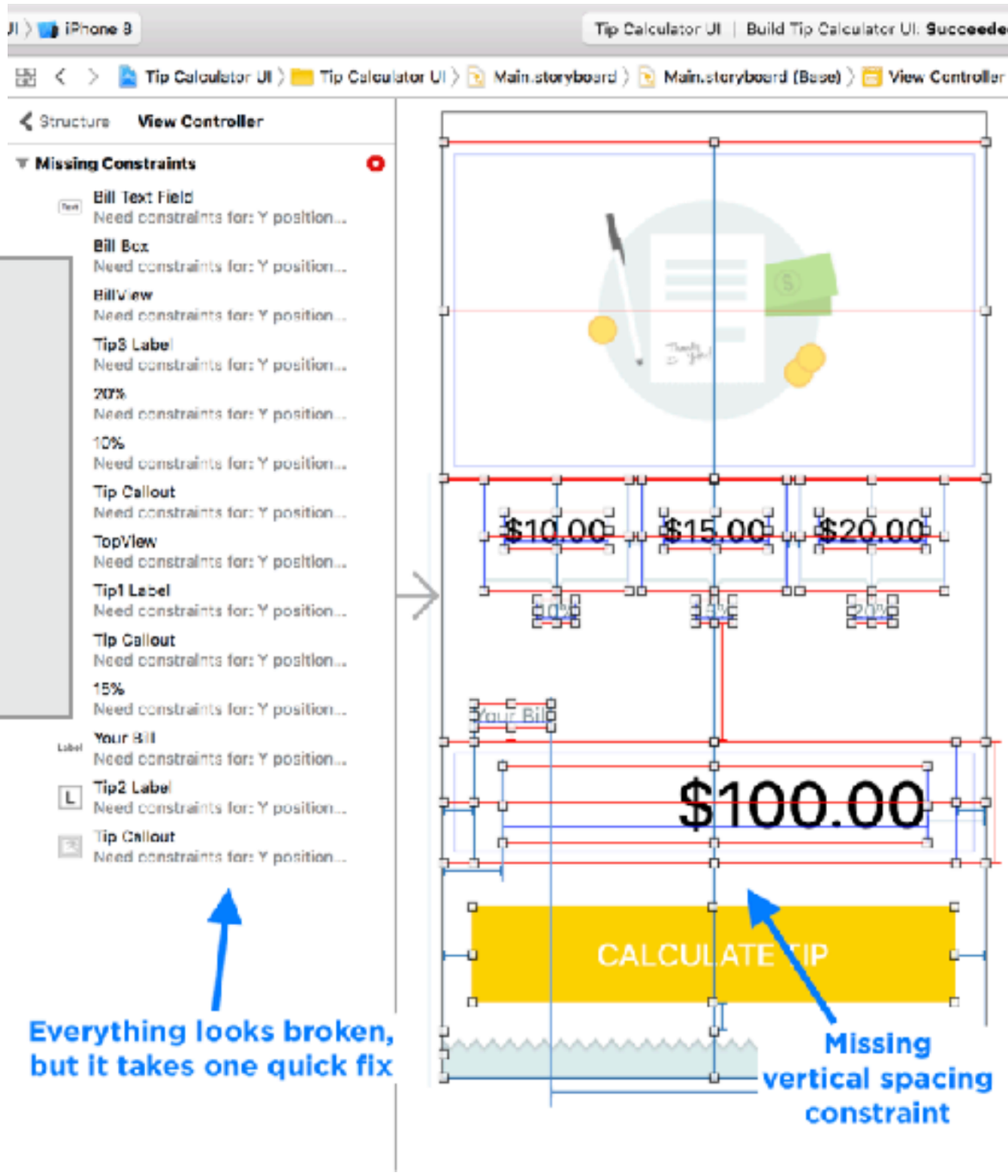
SwiftUI

- **SwiftUI** has a more steep learning curve
- Requires learning a new style of coding for laying out views
- Integrates with UIKit and can be mixed

auto layout with storyboard

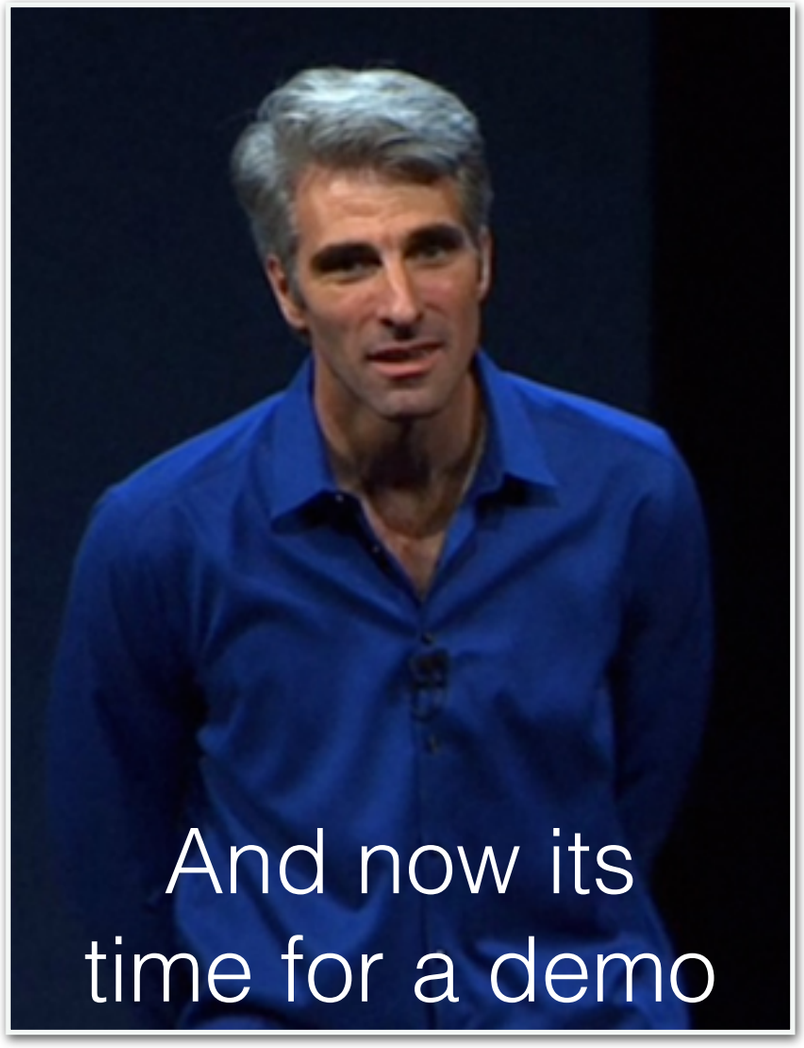


This is the worst part of using UIKit, but necessary



our first app with Xcode

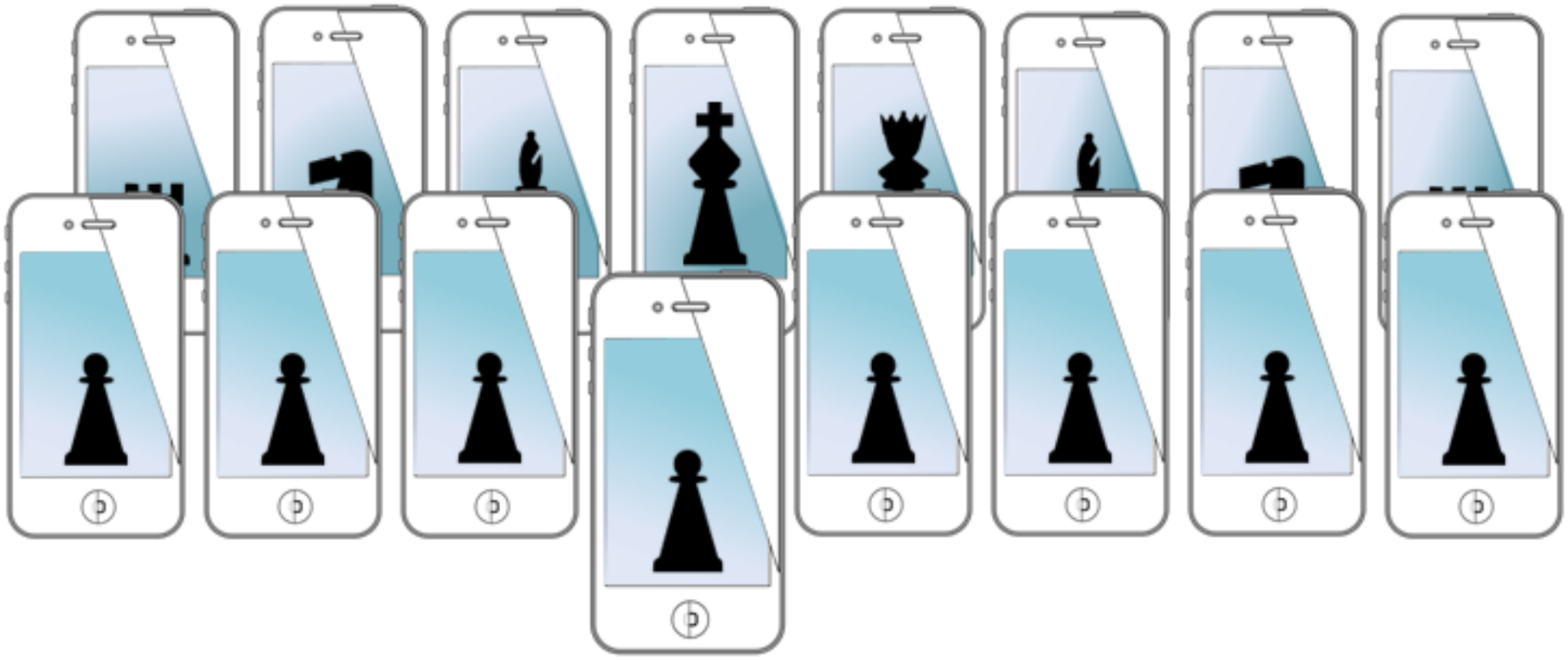
- provides GUI for most git commands
  - commit, branch, push, pull, etc.
- **rarely** is command line needed
- git is great for code but not storyboards
- and some auto layout too!



for next time…

- have teams figured out
- find out how to launch Xcode on your team mac

MOBILE SENSING & LEARNING



CS5323 & 7323

Mobile Sensing and Learning

course introduction

Eric C. Larson, Lyle School of Engineering,
Department of Computer Science, Southern Methodist University