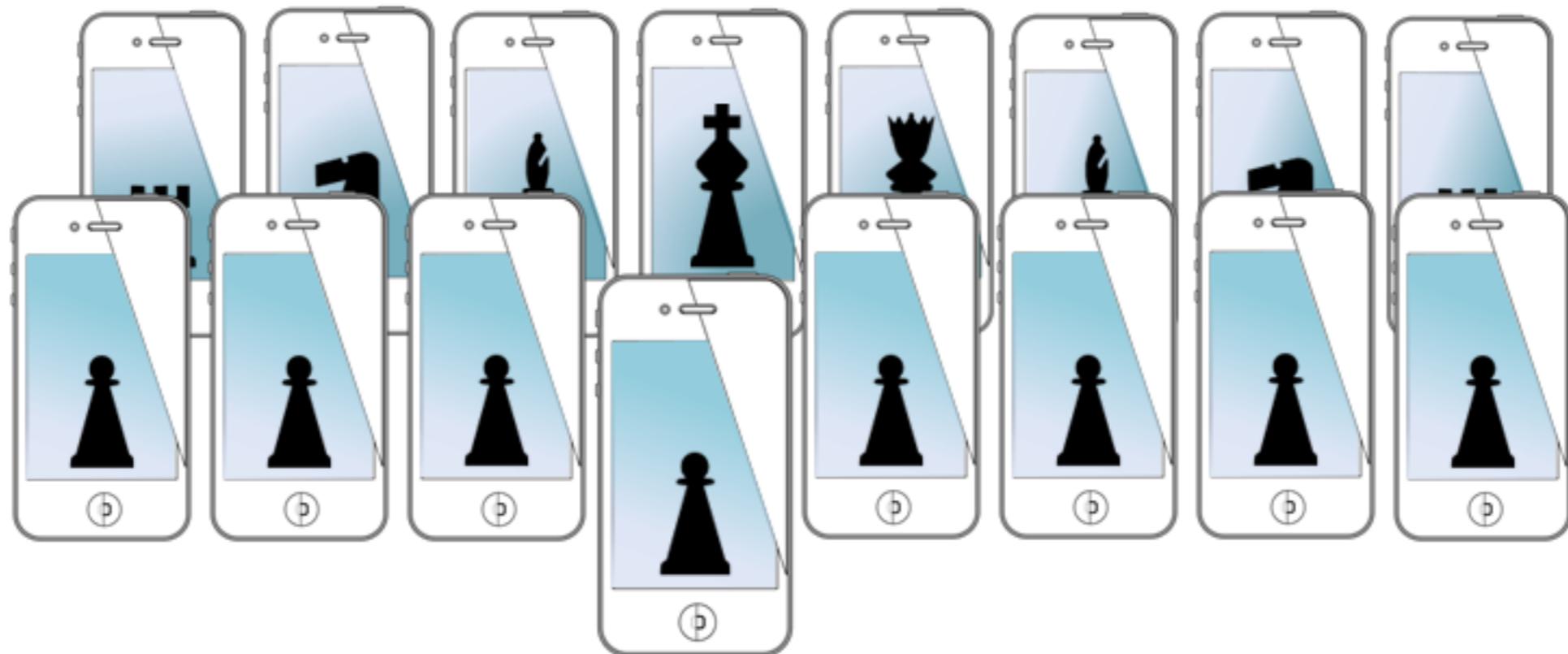


MOBILE SENSING LEARNING



CS5323 & 7323
Mobile Sensing and Learning

mobile design and interaction

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

course logistics

- lab one due at the end of the week!
- how did the flipped module go?
 - any suggestions to improve?

agenda

- mobile HCI
 - design, navigation, and interaction
 - many elements courtesy of apple
 - and some from others

another great resource:

227 pages of bliss, text, pictures, and video
navigable via web interface

iOS Human Interface Guidelines

<https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/>

what makes great UI

- you know it when you see it
- no matter what I tell you, nothing is a hard and fast rule except:
 - keep it simple, clear
 - kill the clutter, display only what is needed
 - use motion/physical metaphors when appropriate

a better slide

deference

never compete
with content

text legible
clarity
background subtle

subtle visual
motion cues

depth

planning

step one

look at core function

step two

add design sparsely

step three

examine assumptions
question every element

planning the app

- **list** potential features of the app
- a recipe app

creating lists

getting recipes

comparing prices

locating stores

annotating recipes

getting and using coupons

viewing cooking demos

exploring different cuisines

finding ingredient
substitutions

planning the app

usually cook at home or prefer ready-made meals

are committed coupon-users or think that coupons aren't worth the effort

enjoy hunting for speciality ingredients or seldom venture beyond the basics

follow recipes strictly or use recipes as inspiration

buy small amounts frequently or buy in bulk infrequently

what do these users want?

who is your target audience?

- **determine** users, list it out

want to keep several in-progress lists for different purposes or just want to remember a few things to buy on the way home

insist on specific brands or make do with the most convenient alternatives

tend to buy a similar set of items on each shopping trip or buy items listed in a recipe

audience: love to experiment with recipes, are often in a hurry, and are thrifty if it doesn't take too much effort

planning the app

- **filter** potential features through your audience

audience: love to experiment with recipes, are often in a hurry, and are thrifty if it doesn't take too much effort

creating lists

getting recipes

comparing prices

Locating stores

annotating recipes

getting and using coupons

viewing cooking demos

exploring different cuisines

finding ingredient substitutions

principles of great design

how to make this slide better?

principles of great design

make it obvious



make it obvious



text entry is awful

- avoid text input at all costs
- e.g., enter your state

solution text

Enter State

solution picker

Select State

GA

HI

ID

IL

IN

IW

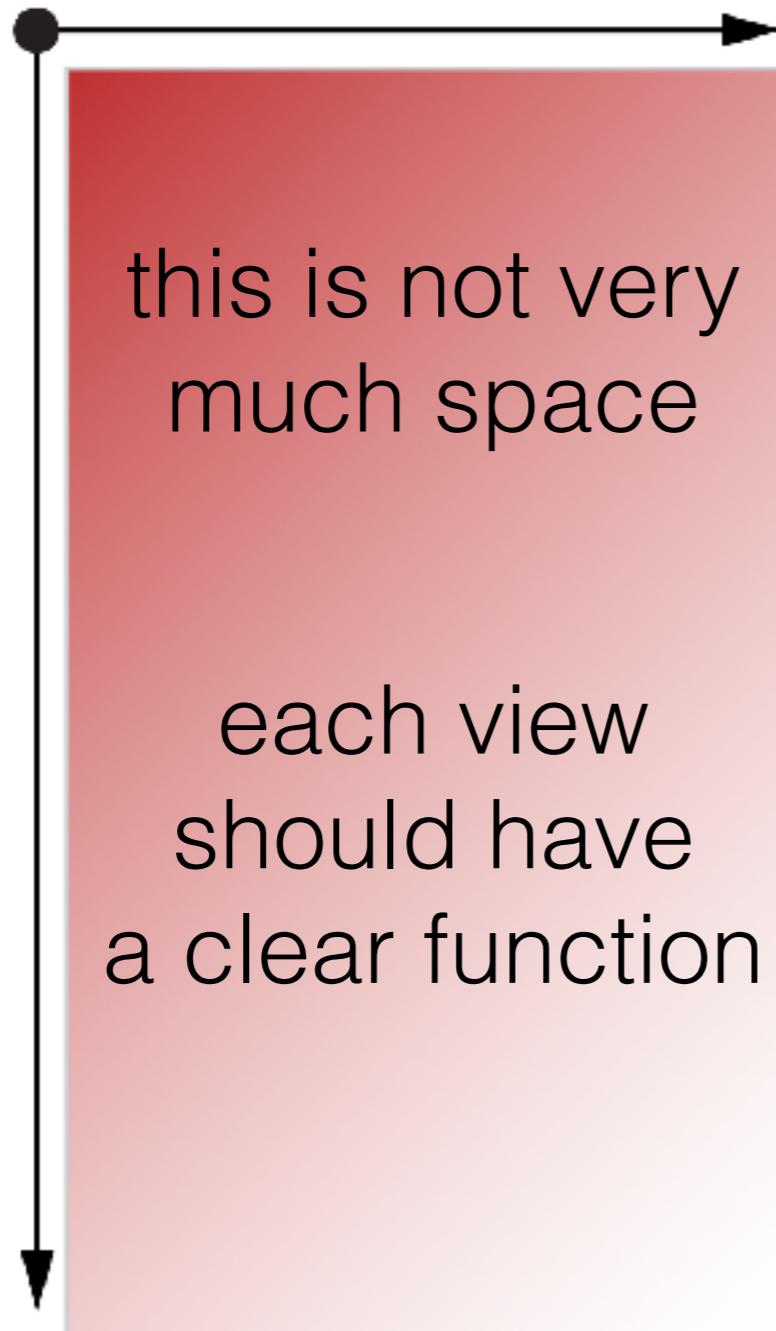
KA

solution get from current location

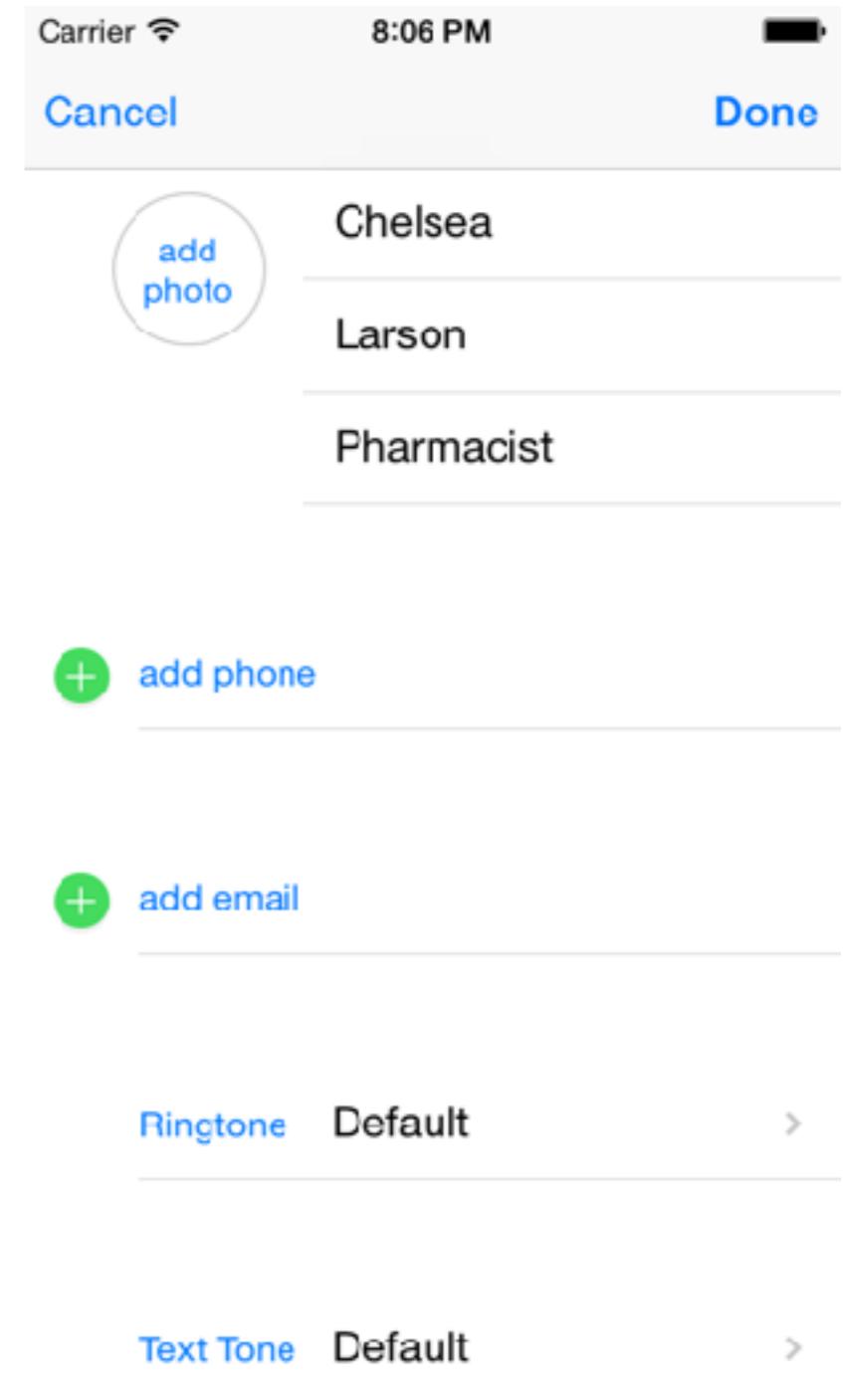
Autofill

layout with care

important



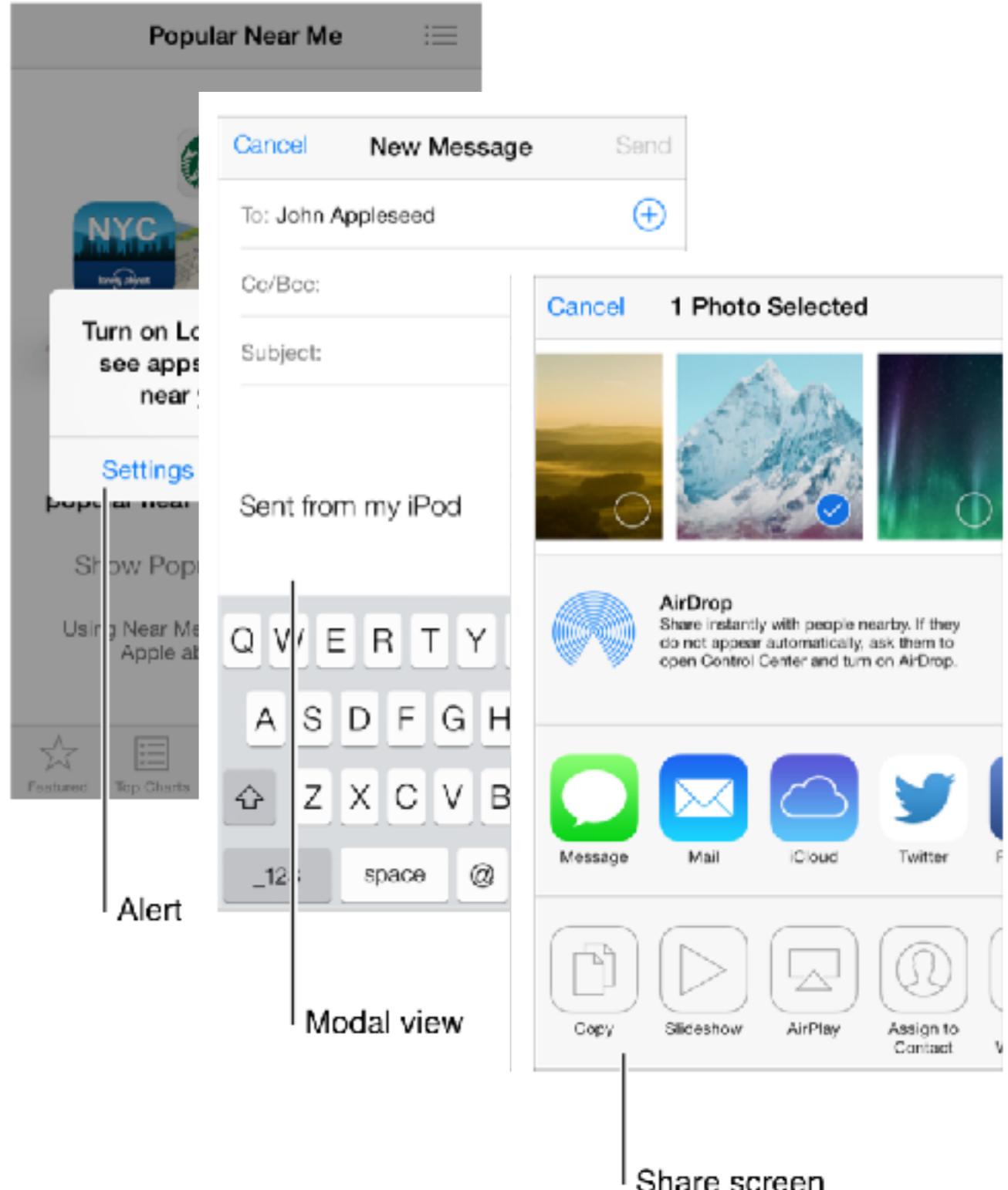
not so important



use modal sparingly

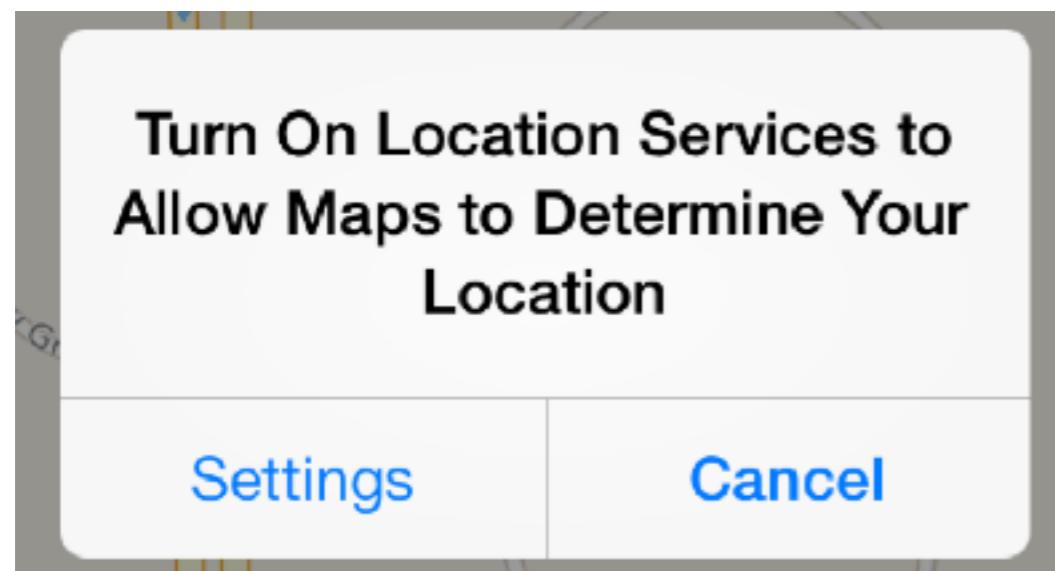
takes over the screen

- prevents other interaction
- use when:
 - critical to capture attention
 - self contained action
 - like sending email
- keep simple, not tied to nav
- provide easy, safe exit



keep alerts succinct

- change wording to be clear and concise
- avoid jargon



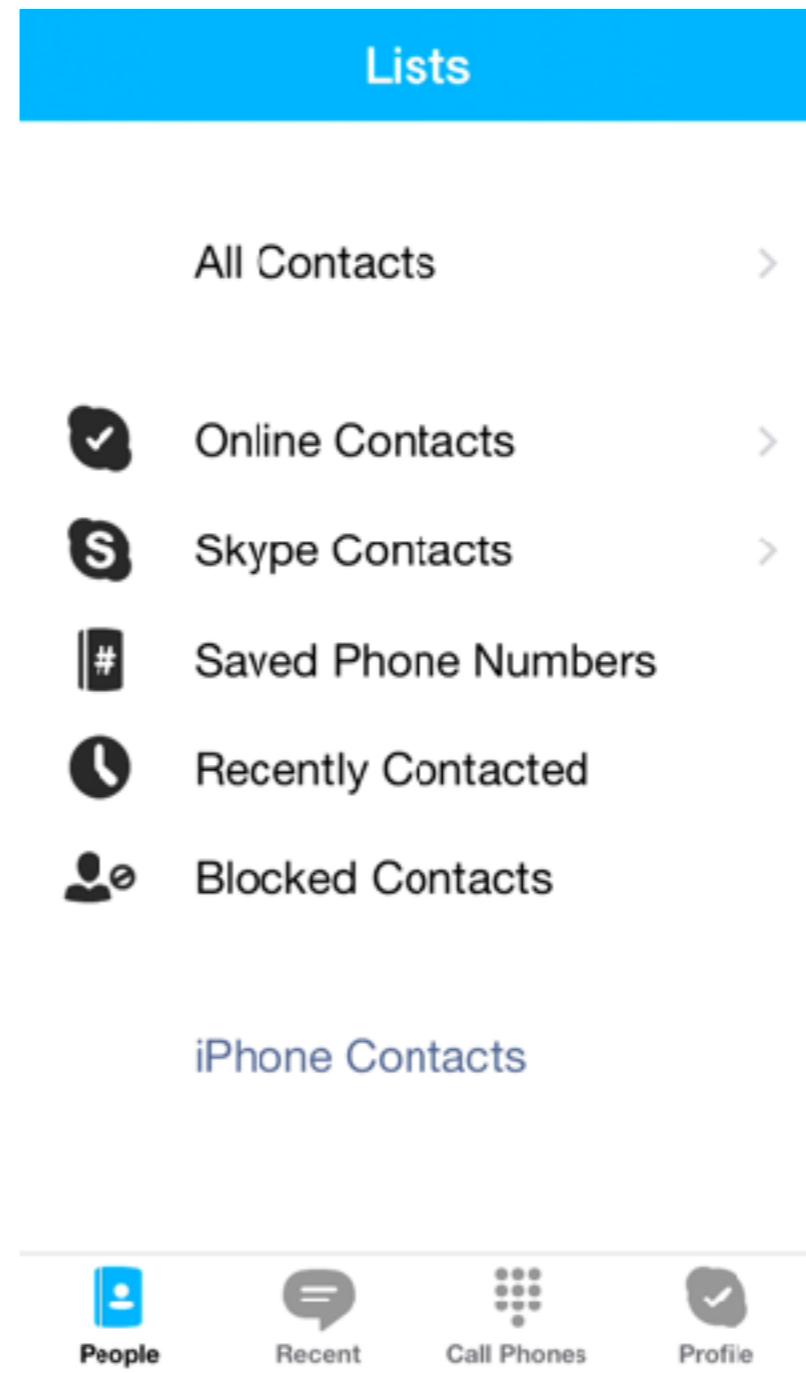
- use verbs for buttons
- title is meaningful
- two choices, safe choice bolded
- full sentences used in explanation
- try to only have a title
- try not to say “we” and “our”

consider no splash

- use the splash screen to give impression of quickness

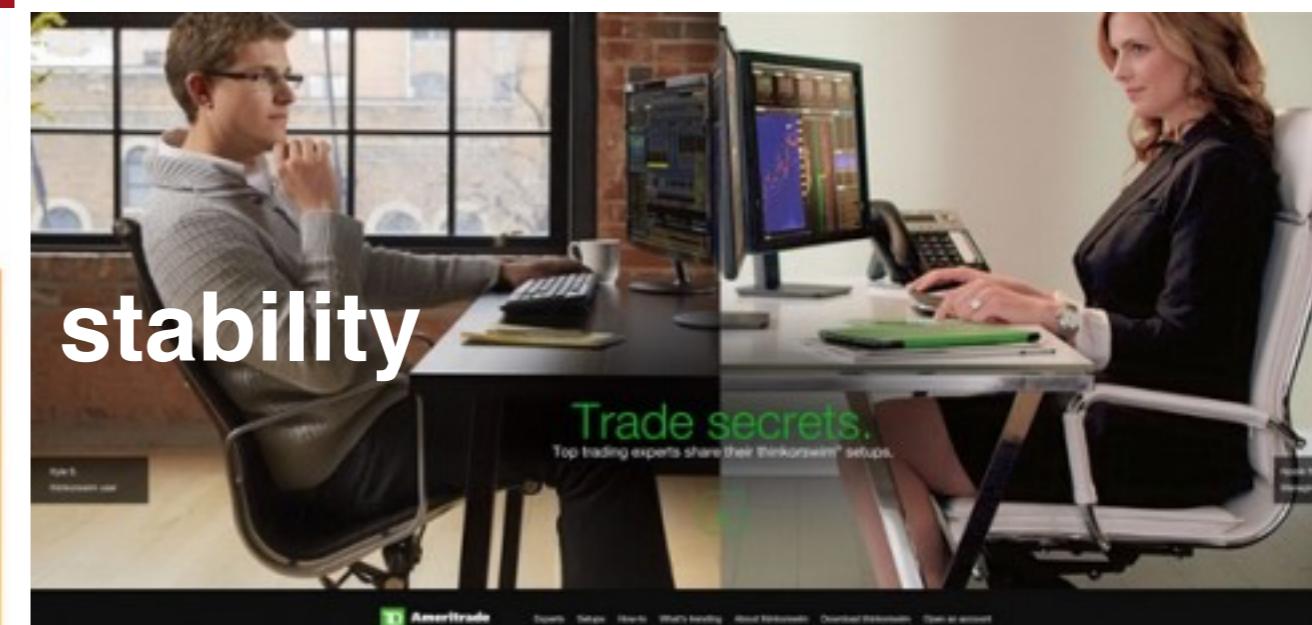
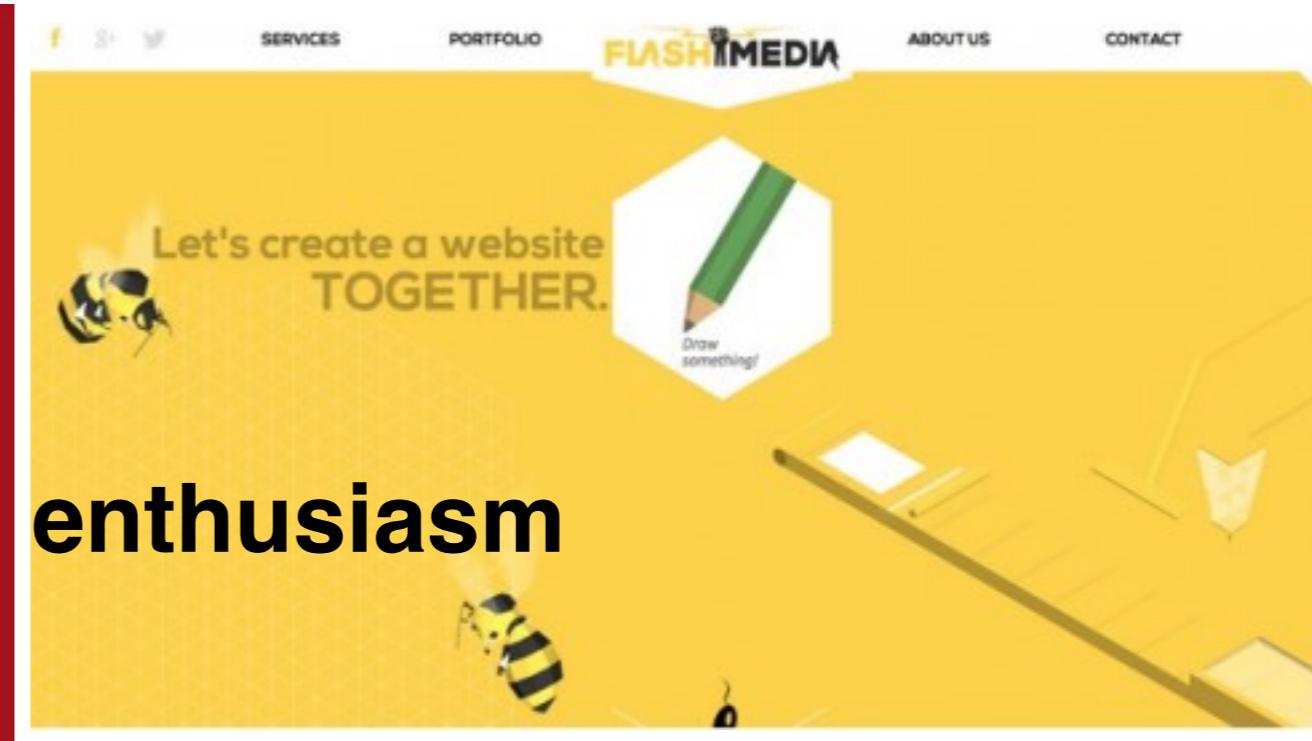


no



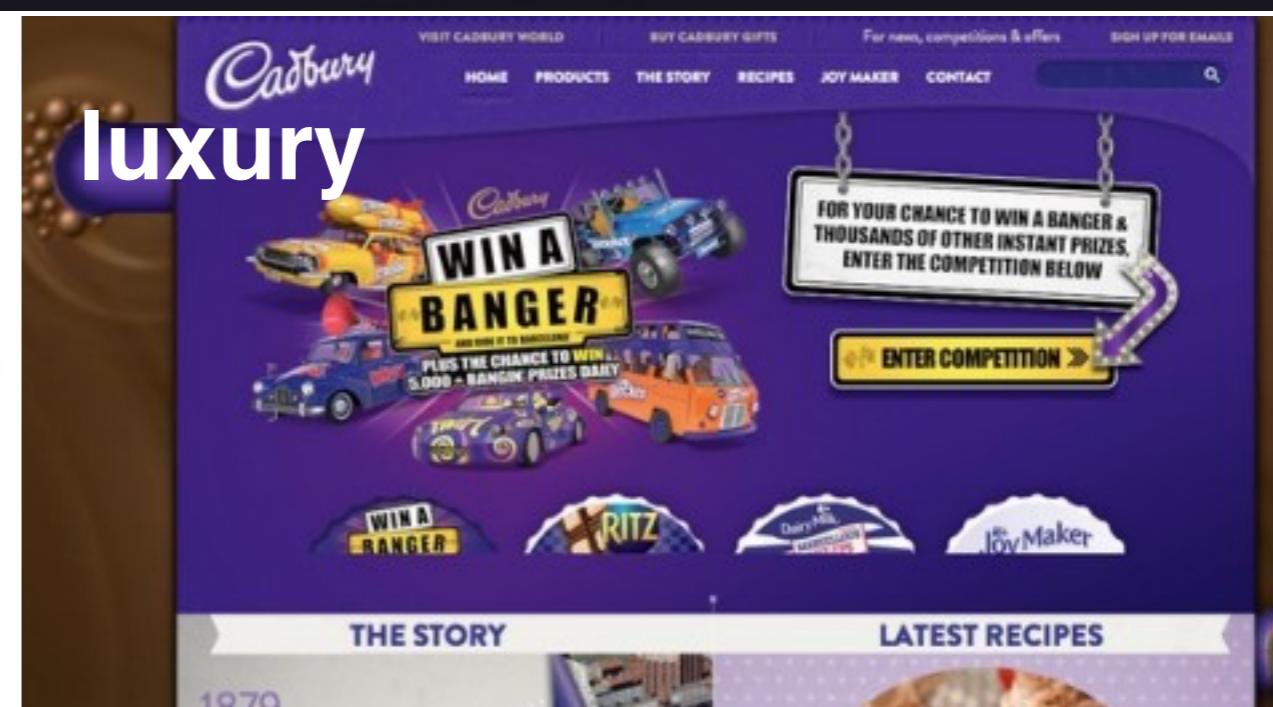
yes

use color “theory”



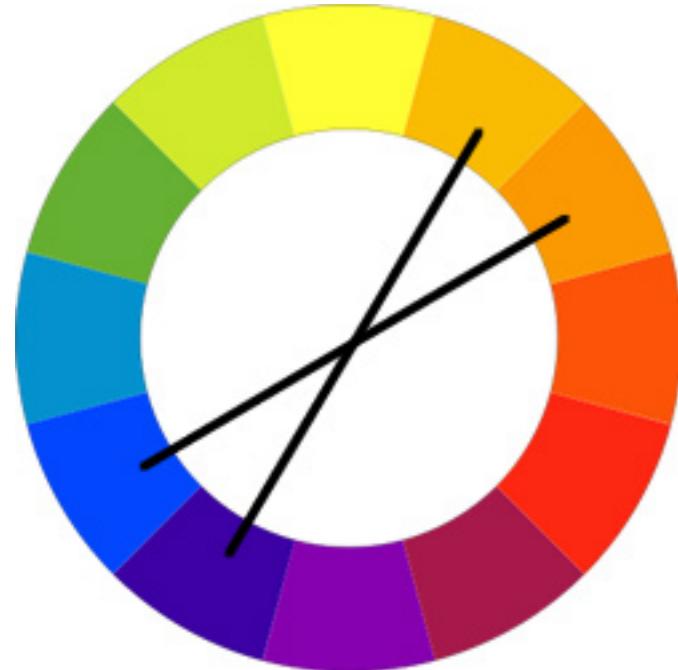
<http://thenextweb.com/dd/2015/04/07/how-to-create-the-right-emotions-with-color-in-web-design/>

use color “theory”



<http://thenextweb.com/dd/2015/04/07/how-to-create-the-right-emotions-with-color-in-web-design/>

use color



The image displays two screenshots of websites demonstrating color usage:

LemonStand (Top Screenshot): The page features a color wheel graphic at the top left. The text "The Only eCommerce Platform for Custom Online Stores" is displayed above a "Try It For Free" button. Below the button, there are three bullet points: "Easily customize anything, front-end or back", "Generate income by selling your custom modules", and "Support directly from our software engineers". To the right is a screenshot of a computer monitor showing a complex code editor with multiple tabs open.

National Multifamily Housing Council (Bottom Screenshot): This is a landing page for "APARTMENTS". The title "WE LIVE HERE" is centered. Below it is a paragraph: "In communities across the country, apartments work – helping people live in a home that's right for them. And demand continues to grow. Learn how apartments create communities and contribute to the economy." The background features stylized illustrations of buildings, trees, and clouds. At the bottom, there is a navigation bar with links like "Start", "We Build More", "We Work More", "We Spend More", "Start Over", and logos for NMHC and NAA.

<http://thenextweb.com/dd/2015/0>

use color theory



Analogous

mono-themed, but elegant
easy to select one other contrast color or highlights without
deviating from theme

match your
app icon
and
UI palette

Silverback 2.0
Guerrilla usability testing software for designers and developers

- Capture screen activity
- Video the tester's face
- Record the tester's voice
- Add chapter markers on-the-fly
- Control recording with the remote
- Export to Quicktime

Features in 2.0 include

Preview	Batch Export
Watch sessions within Silverback	Save selected sessions, tasks, highlights or projects in one go
Tasks & Highlights	Performance
Set tasks and mark noteworthy moments within a session	Faster export, better usability

Download **Buy NOW** \$69.95 FREE upgrade for existing users

What does Silverback do?

<http://thenextweb.com/dd/2015/04/07/how-to-create-the-right-emotions-with-color-in-web-design/>

Apple says... design is...

formatting



design for touch



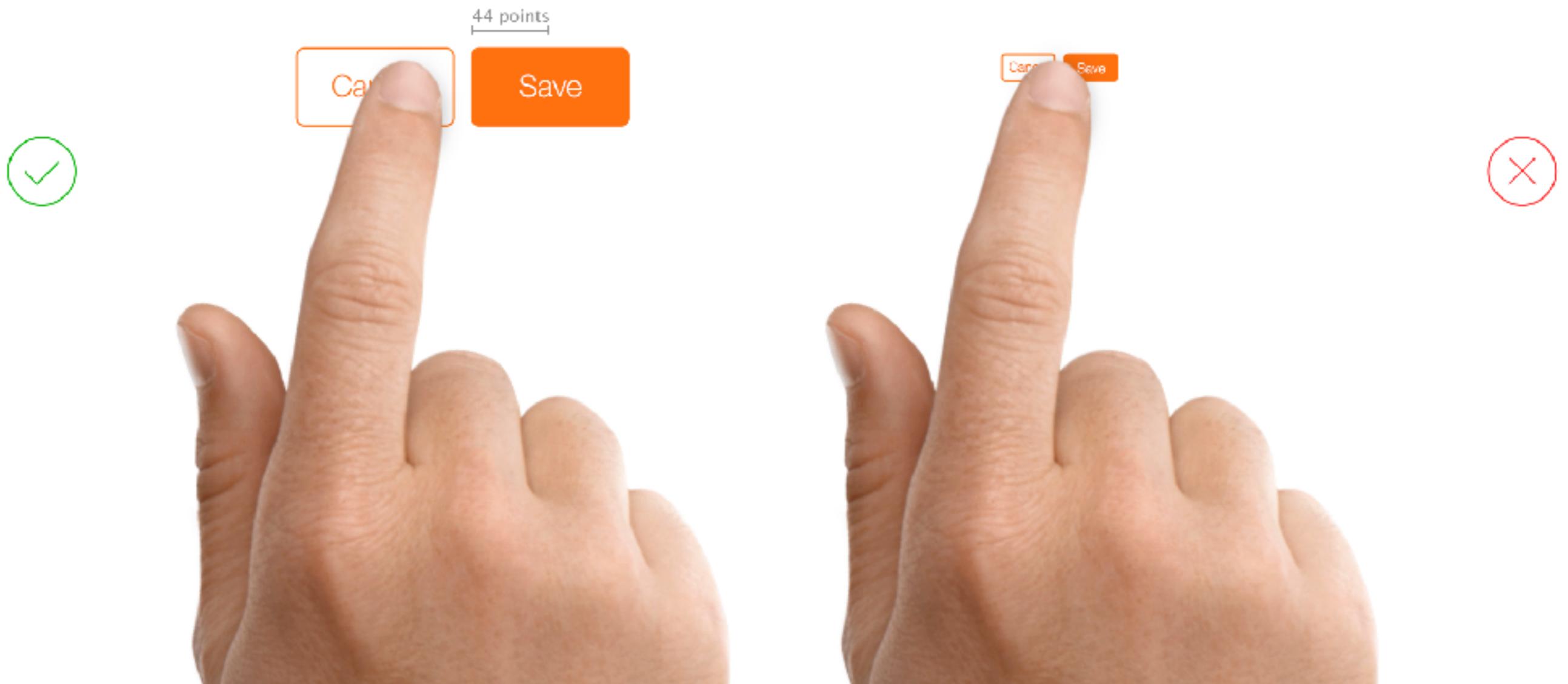
Date	October 11, 2013	4:00 PM
Tue Oct 8	1	57
Wed Oct 9	2	58
Thu Oct 10	3	59 AM
Fri Oct 11	4	00 PM
Sat Oct 12	5	01
Sun Oct 13	6	02
Mon Oct 14	7	03

Time: : AM

Date:

October						
S	M	T	W	Th	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

design for taps



legible text

Heading

Sub-Headline



Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend egestas nisl vehicula nec. Nullam varius est dui, nec accumsan lectus posuere ut. Nullam viverra purus laoreet euismod tempor.

Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend egestas nisl vehicula nec. Nullam varius est dui, nec accumsan lectus posuere ut. Nullam viverra purus laoreet euismod tempor.

Heading

Sub-Headline

Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend egestas nisl vehicula nec. Nullam varius est dui, nec accumsan lectus posuere ut. Nullam viverra purus laoreet euismod tempor.

Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend egestas nisl vehicula nec. Nullam varius est dui, nec accumsan lectus posuere ut. Nullam viverra purus laoreet euismod tempor.

Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend egestas nisl vehicula nec. Nullam varius est dui, nec accumsan lectus posuere ut. Nullam viverra purus laoreet euismod tempor.



high contrast

Heading

Sub-Headline



Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend egestas nisl vehicula nec. Nullam varius est dui, nec accumsan lectus posuere ut. Nullam viverra purus laoreet euismod tempor.

Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend.

Heading

Sub-Headline



Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend egestas nisl vehicula nec. Nullam varius est dui, nec accumsan lectus posuere ut. Nullam viverra purus laoreet euismod tempor.

Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend.

negative space

Heading

Sub-Headline



Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend egestas nisl vehicula nec. Nullam varius est dui, nec accumsan lectus posuere ut. Nullam viverra purus laoreet euismod tempor.

Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend.

Heading

Sub-Headline



Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend egestas nisl vehicula nec. Nullam varius est dui, nec accumsan lectus posuere ut. Nullam viverra purus laoreet euismod tempor.

Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend.

design for retina



organize



Edit	
Coffee	28 g >
Grain Size	~113.3 µm >
Water	1241 ml >
Temperature	103°C >
Time	223 s >
Serving	310.25 ml >
Metric	English



coffee: 28 g. Edit grain size:
~113.1 µm Edit water: 1241
ml Edit temp: 103° Edit
time: 223 s. Edit serving:
310.25 ml Edit

Metric English
Celsius Fahrenheit

alignment



Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare*
Curabitur semper vitae urna ac adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus.



**Ornare imperdiet blandit lectus. Morbi tristique*

Continue

Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare* Curabitur semper vitae urna ac tempus.



**ornare imperdiet blandit lectus.
Morbi tristique*

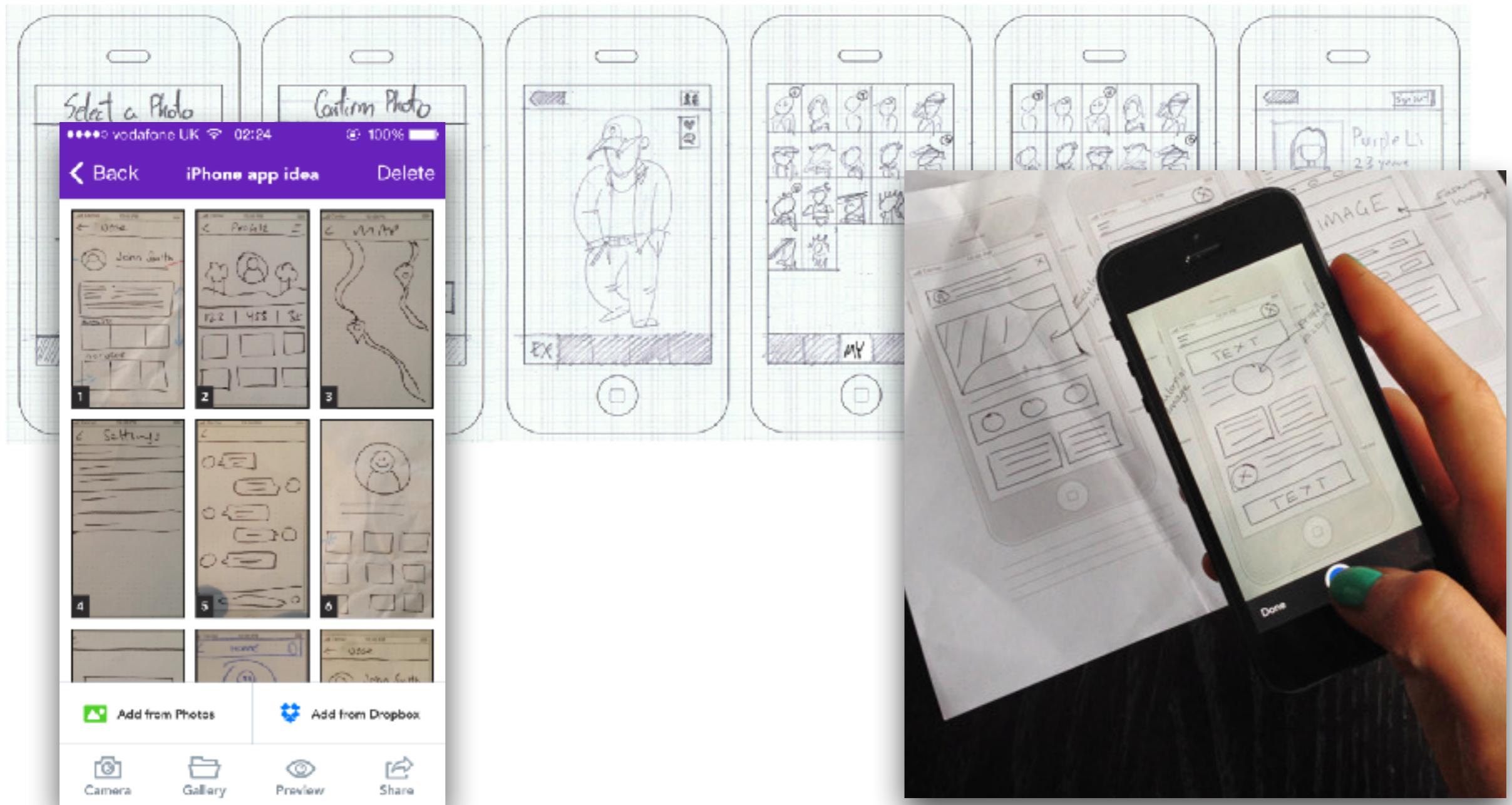
Continue



Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus.
Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus.

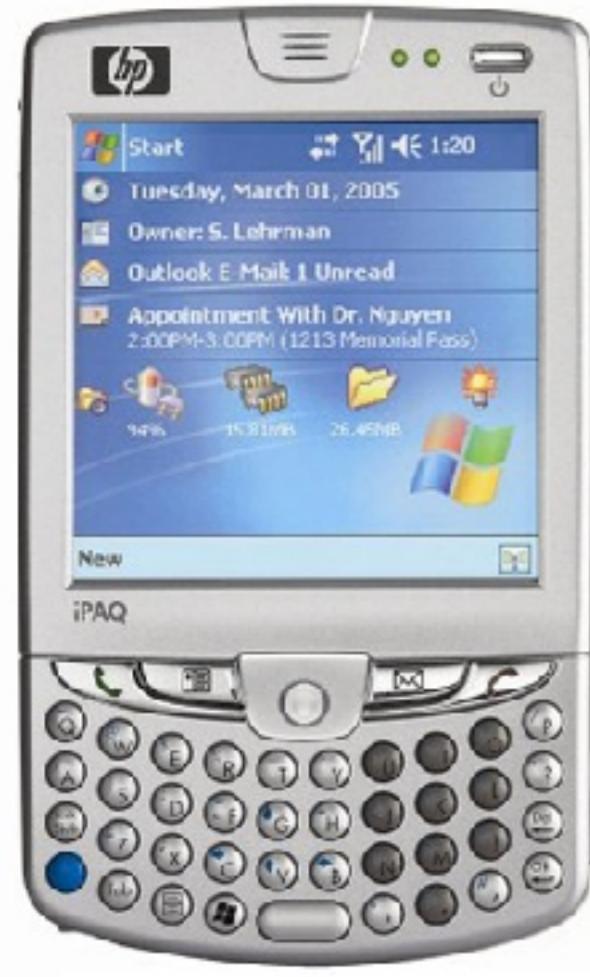
get feedback early

paper prototypes



flat design

- HCI was built on the concept of metaphor since its inception
 - called skeuomorphism
 - mobile HCI breaks all these metaphors



flat design

- no need for visual metaphor if intuitive
- make minimal and efficient
- color is king



flat design

- main principles started in advertising
- the swiss style and minimalist style



pure flat design

- minimal typography
 - Helvetica (Neue) / SF
 - text size gives importance
- colors are vibrant
 - set against light/dark
- no drop shadows
- no lightpoint gradients
- depth is for orienting only, never the UI elements



R 255
G 59
B 48

Red

R 255
G 149
B 0

Orange

R 255
G 204
B 0

Yellow

R 76
G 217
B 100

Green

R 90
G 200
B 250

Teal Blue

R 0
G 122
B 255

Blue

R 88
G 86
B 214

Purple

R 255
G 45
B 85

Pink

pure flat design?

eric c. larson

HOME PUBLICATIONS TALKS TEACHING CV

COURSES OFFERED

- Mobile Sensing and Learning
- Machine Learning in Python
- Machine Learning and Neural Networks
- Ubiquitous Computing
- Introduction to Data Mining
- Computer Science Seminar

CSE5323 & CSE7323 - MOBILE SENSING AND LEARNING

MOBILE SENSING LEARNING

COURSE DESCRIPTION

This class will equip students with the practical skills necessary to develop mobile applications able to take advantage of the myriad of sensing, machine learning, and control capabilities that modern smartphones offer. The course focuses on interfacing with the hardware of the phone and inferring high level information from the sensors streams. Particular focus will be placed upon efficiently analyzing and controlling hardware peripherals on third party hardware, such as an embedded micro-controller or peripheral such as Google Glass. This third-party hardware platform will interface with the mobile platform and allow students to integrate realtime control/automation with the sensing learned earlier in the semester. Assignments will use both objective C and C++ programming languages, on the iOS platform. Feel free to contact the instructor at eclarson@smu.edu if you have any questions.



Associate Professor
Computer Science
Bobby B. Lyle School of Engineering
Southern Methodist University

twitter: @ec_larson
email: eclarson@smu.edu

CS Office: 451 Caugh Hall
Lyle School of Engineering
Caruth Hall
3145 Dyer Street, Suite 445
Dallas, TX 75200

SMU UbiComp Lab:
Johnson Square 119

eric c. larson

HOME PUBLICATIONS TALKS TEACHING CV

Filter: | On my Google Scholar page.

2019

GROUP BY Year Pub Type None

Y-AR
2018 (6)
2018 (2)
2017 (4)
2016 (2)
2016 (4)
2014 (3)
2013 (6)
2012 (8)
2011 (8)
2010 (9)
2009 (2)
2008 (4)
2007 (1)

Keyboard Snooping From Mobile Phone Arrays With Mixed Convolutional and Recurrent Neural Networks
Tyler Gialanella, Travis Siems, Elena Smith, Erik Gabrielsen, Ian Johnson, Mitchell A Thornton, Eric C Larson
IMWUT 2019
Keywords: side channel, security, convolutional neural networks, mobile phones

Download: [pdf] Export: [Citation]

Why Deep Knowledge Tracing Has Less Depth Than Anticipated
Xinyi Ding, Eric Larson
Proceedings of EDM 2019
Keywords: knowledge tracing, recurrent neural networks

Download: [pdf] Export: [Citation]

Computer Science Education: Fueling Tomorrow's Technology Growth
Fred Chang, Eric Larson, Mark Fontenot
Georgetown Journal of International Affairs
Keywords: security, computer science education

Download: [pdf] Export: [Citation]

EduAware: Using Tablet-Based Navigation Gestures to Predict Learning Module Performance
Xinyi Ding, Eric C Larson, Amanda Doyle, Kevin Donahue, Radhika Rajgopal, Eric Ding
Interactive Learning Environments
Keywords: mobile computing, cancer, mobile learning assessment

Download: [pdf] Export: [Citation]



Get more out of Google by using Google+

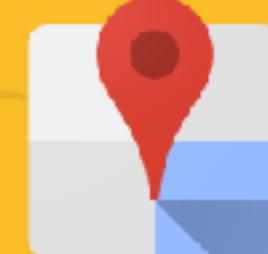
Whether you're using Gmail, Search, YouTube, Maps, or Blogger, Google+ gives you new ways to share the right things with the right people.

[Upgrade to Google+](#)



Gmail »

Video Chat with up to nine people at once; find emails from the people you care about; share photos effortlessly and more.



Maps »

Send directions and share the places you love with just the right people, right from Google Maps.



News »

Get the latest from your favorite authors, and see the articles your friends are sharing right in Google News.



Search »

Search everything on the web, plus your photos, posts from your circles, and the things your friends have shared with you.



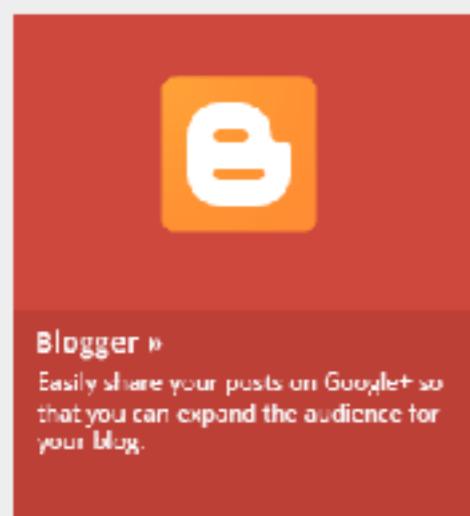
Earth »

Share images of the places you visit in Google Earth to just the right people using Google+.



Youtube »

See which videos your friends are sharing and watch any video with up to nine friends at once.



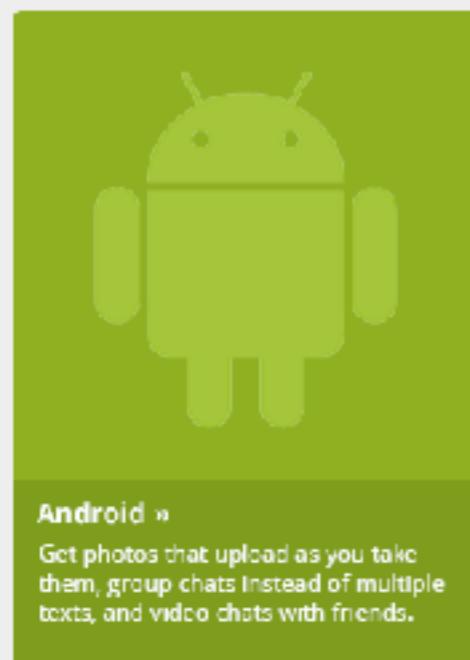
Blogger »

Easily share your posts on Google+ so that you can expand the audience for your blog.



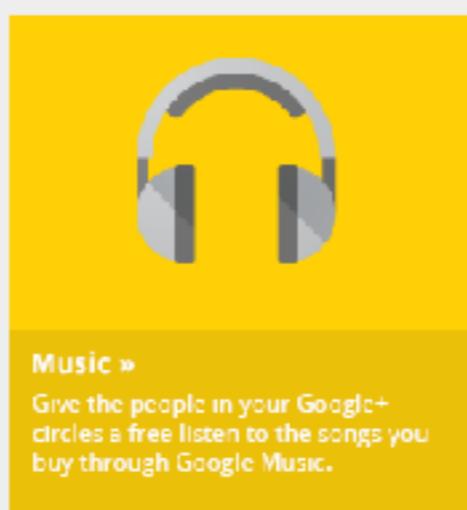
Chrome »

Recommend sites, share pages, and always stay up-to-date with the people in your Google+ circles.



Android »

Get photos that upload as you take them, group chats instead of multiple texts, and video chats with friends.



Music »

Give the people in your Google+ circles a free listen to the songs you buy through Google Music.

defining your style

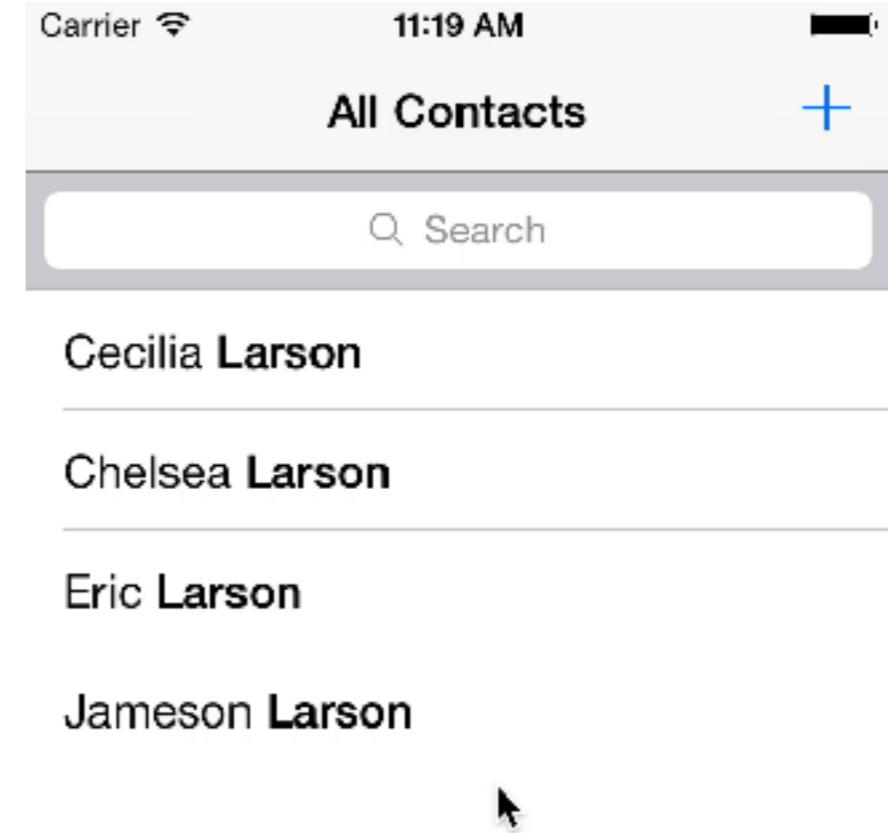
know your app: basics

style basics

design is flat

- color exposes purpose
- negative space
- subtle bordering when needed

- no shadows or bordering
- text filled minimally and descriptive
- **color** conveys interaction possible
- borders when ambiguous



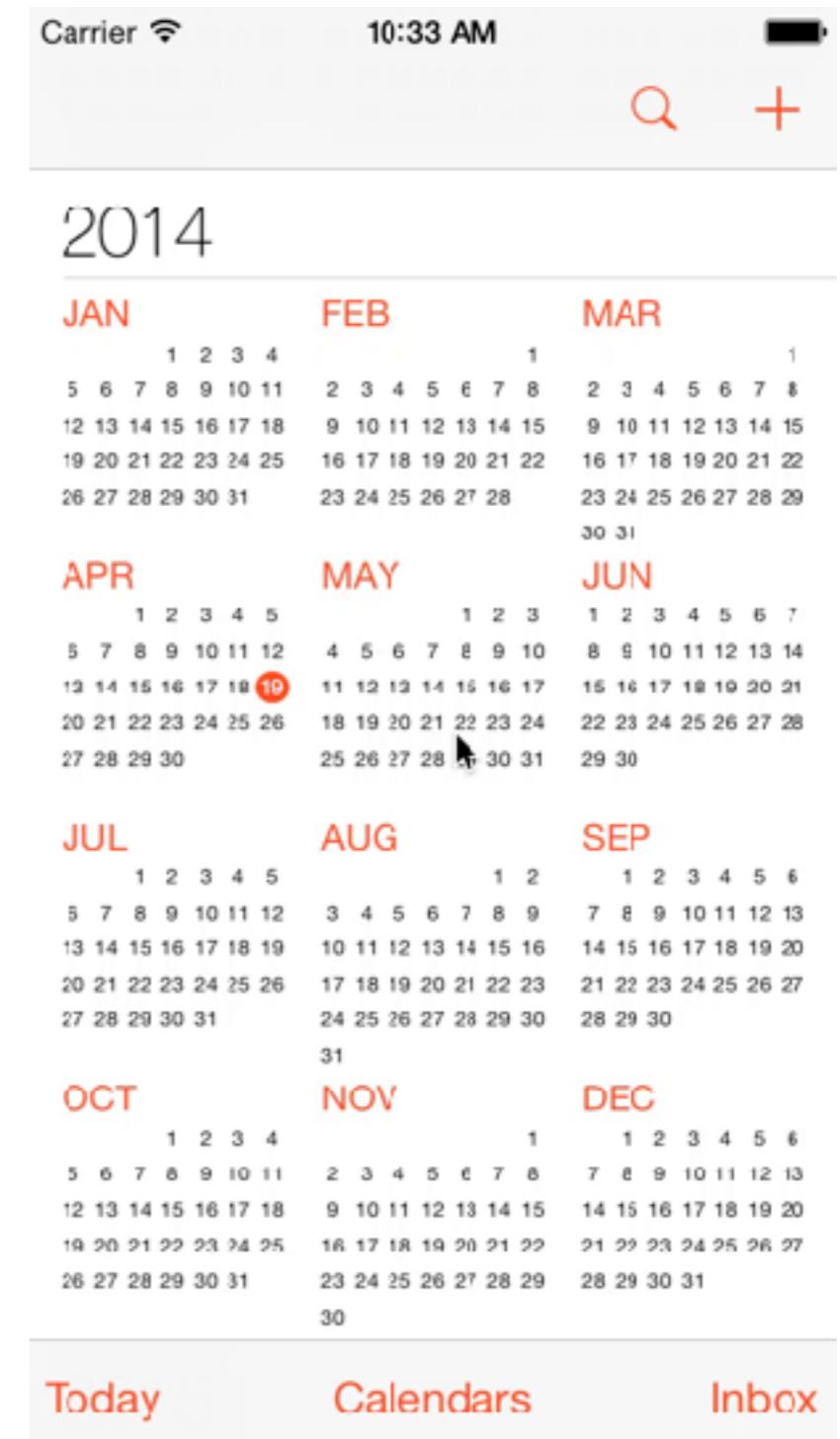
know your app: productivity

productivity apps

enables detailed manipulation

- organizing
- adding and subtracting
- drill down for detail

- depth conveys hierarchy
- transition motion orients user
- detail view takes over the screen
- manipulation tabs change with depth



know your app: utility

utility

simple task, minimal input

- highly visual
 - enhanced display of info
 - no hierarchy
 - glance-able
-
- entire screen is used
 - navigation is flat
 - input is exploratory
 - no elements are in competition

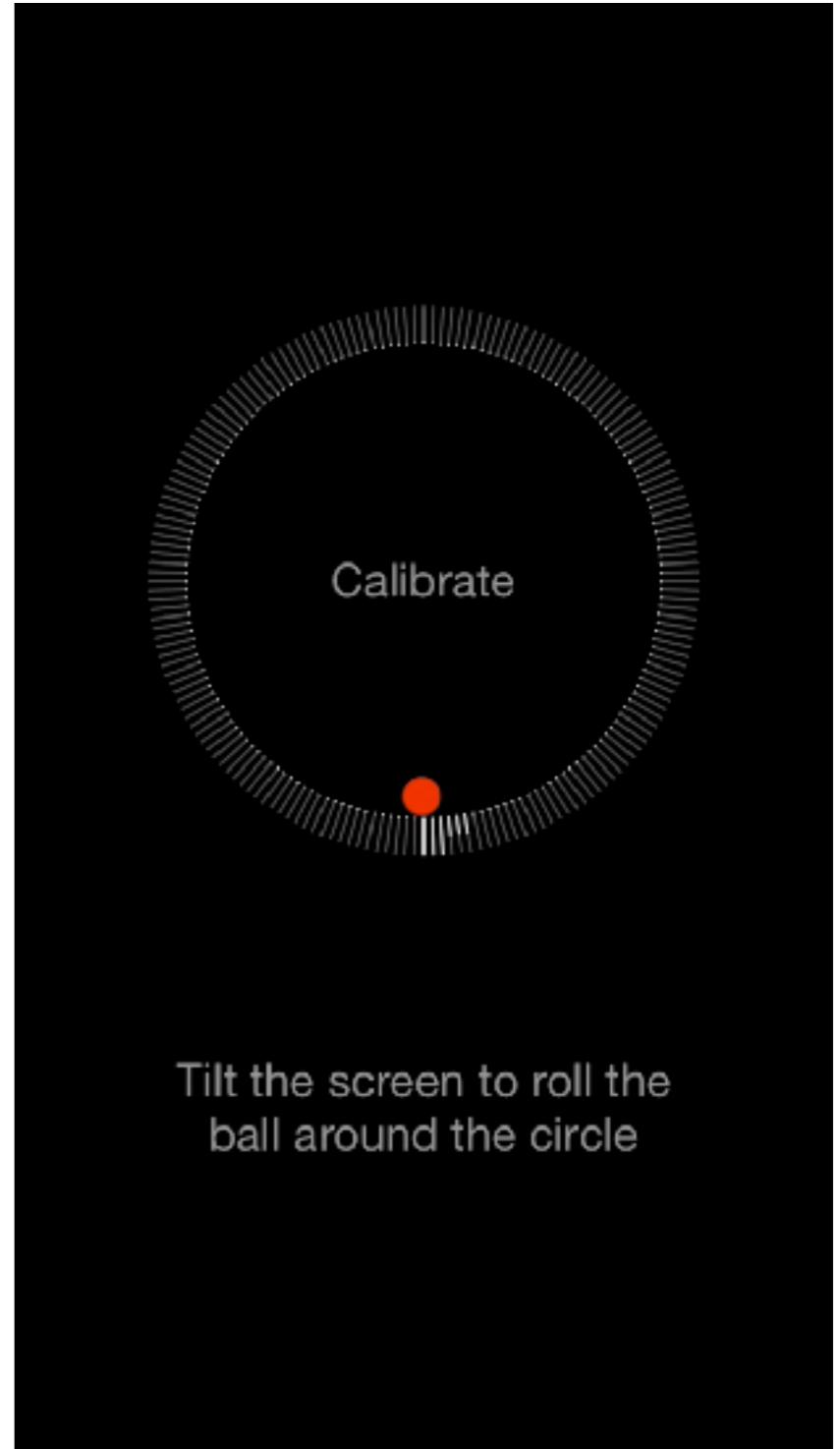


know your app: immersion

immersive

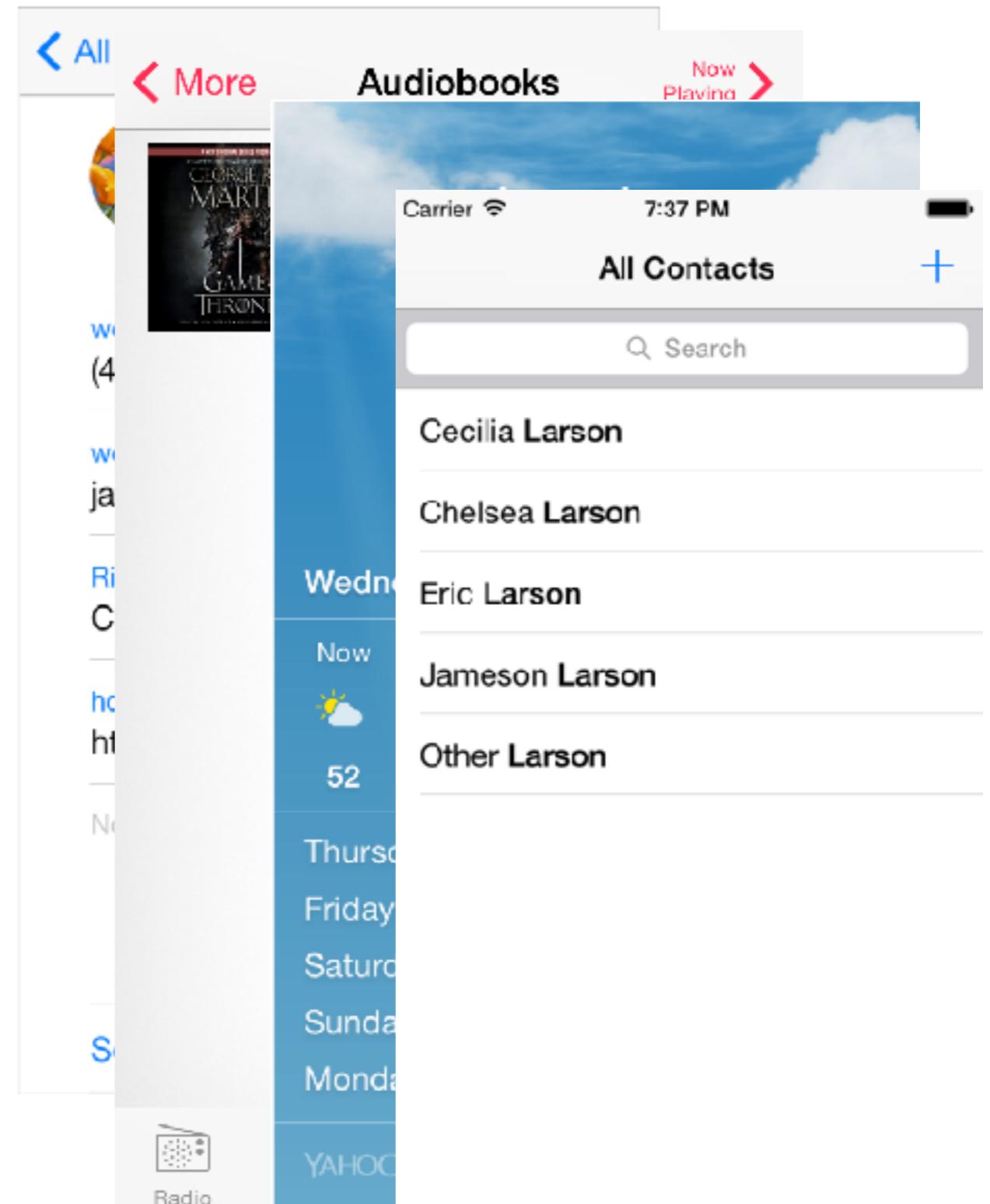
high interaction, visual experience

- hide UI elements
- nonstandard controls
- information centered on story, gameplay, experience



navigation is orientation

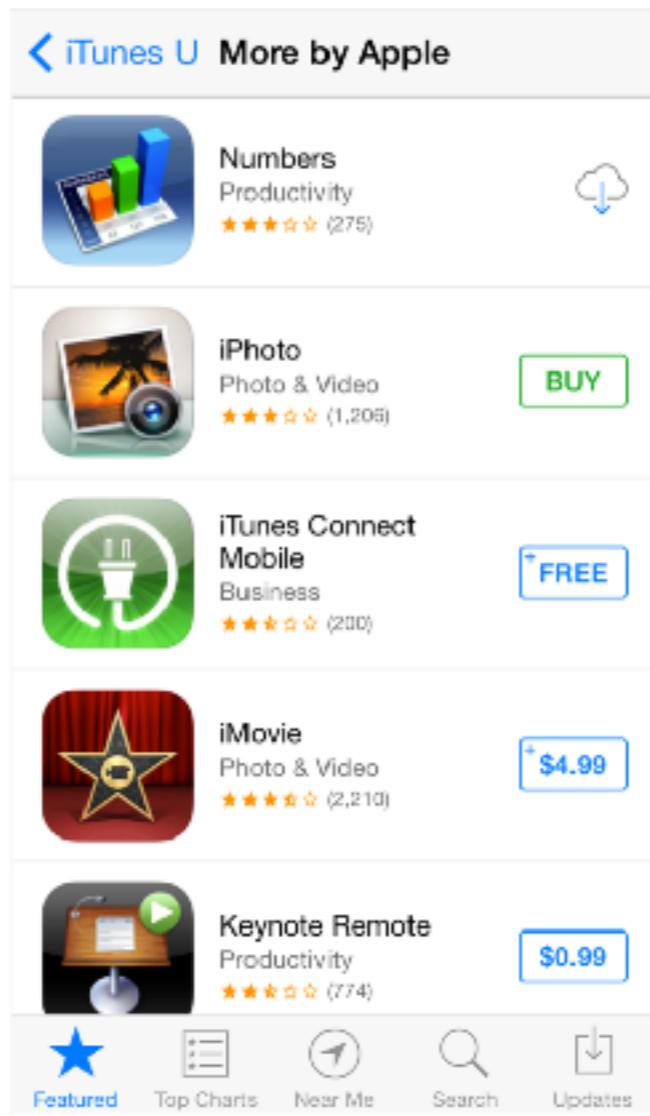
- the user must always know **where they are** in the app
- the **navigation bar** is the most understandable form of hierarchy, even when using animated hierarchy
- **tab bar** is for parallel content (peers)
 - each peer is different in function
- **page control** is for identical views, with new content
- **table view** is for master/detail
- only have **one way** of navigating to a view



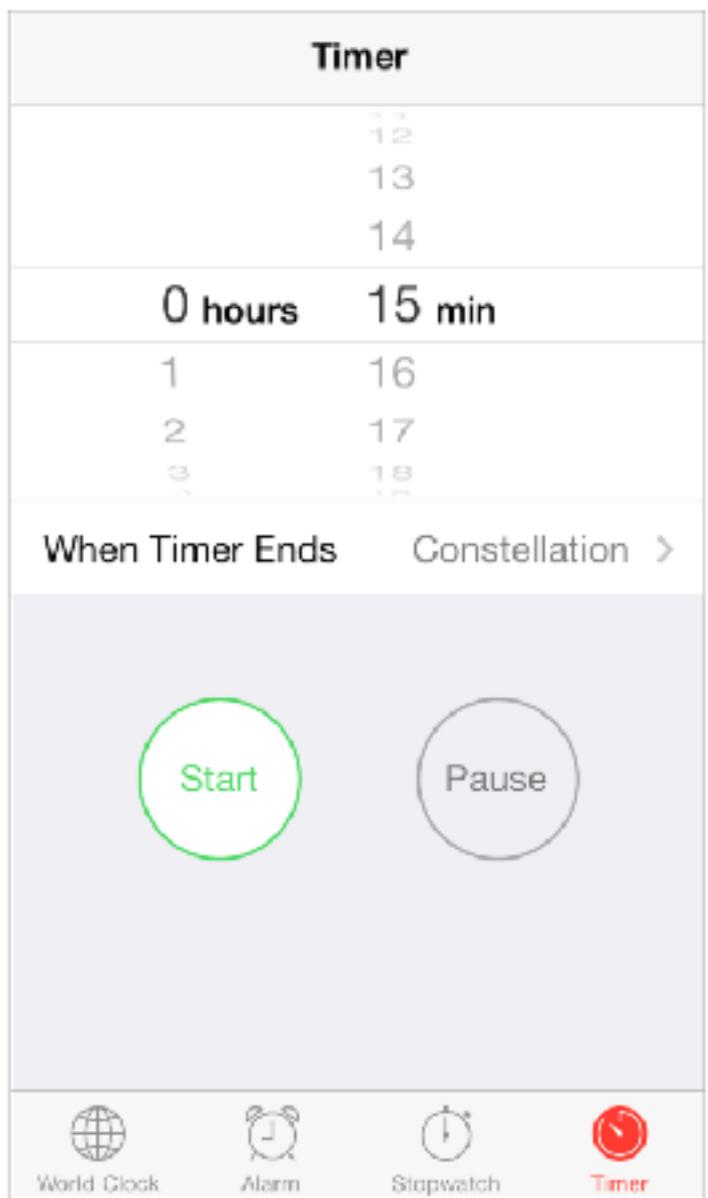
button design

- embrace the borderless
- until you need a border

distinguish
“tap button”
from “tap row”



most
important
interaction



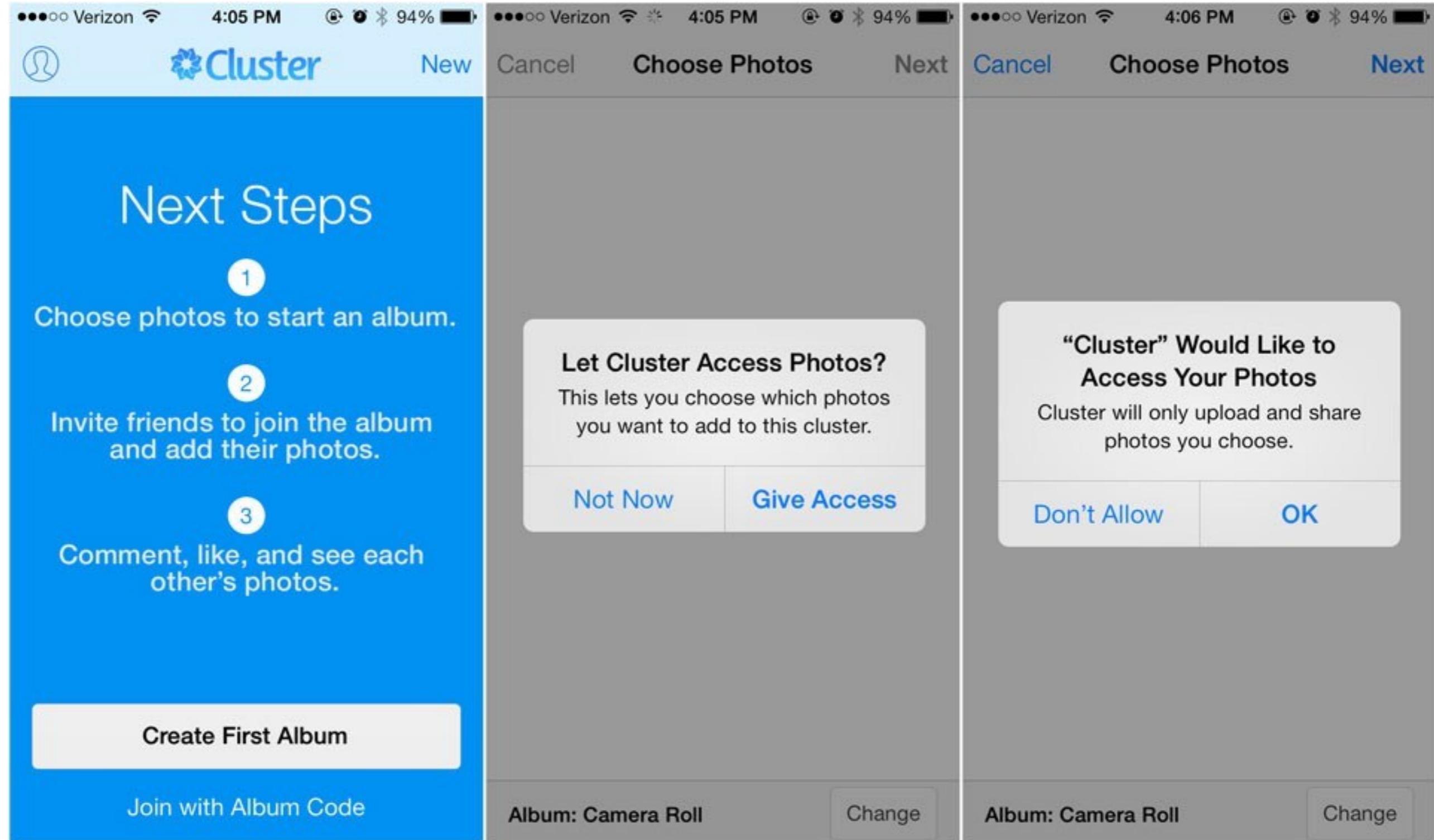
principles summary

- stay consistent
- use design elements that don't distract
 - for a game, engagement is king
 - for productivity, keep animation subtle and quick
- visible feedback and direct manipulation
- use metaphor to promote intuitive interaction
 - flick / tap / pinch
- give the user the control
 - the user always has control of their information

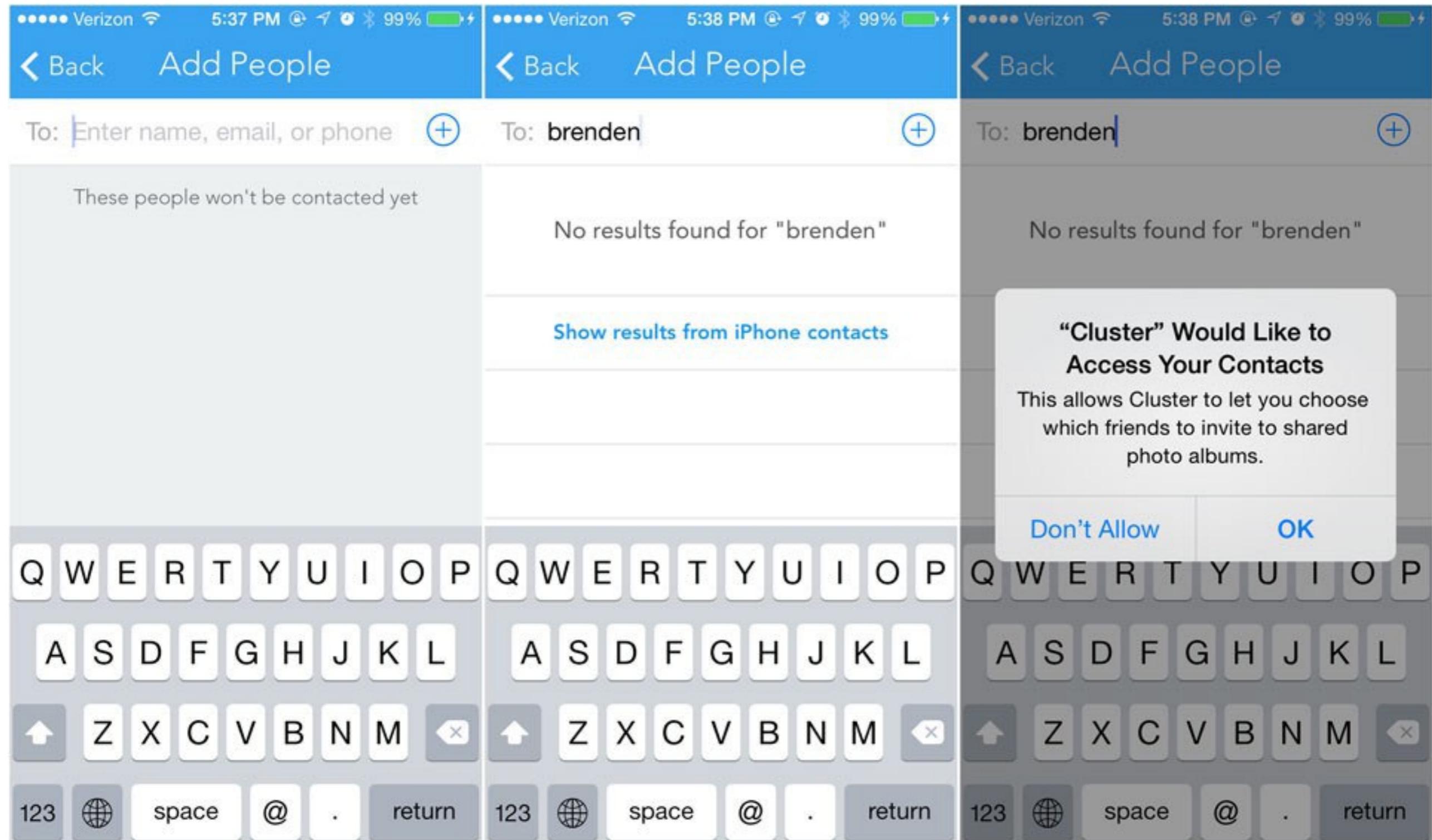
user control

- don't ask until you need it
- make sure the user knows the tradeoff
- use “benefit->explanation”
 - ask twice, showing the benefit in your own words

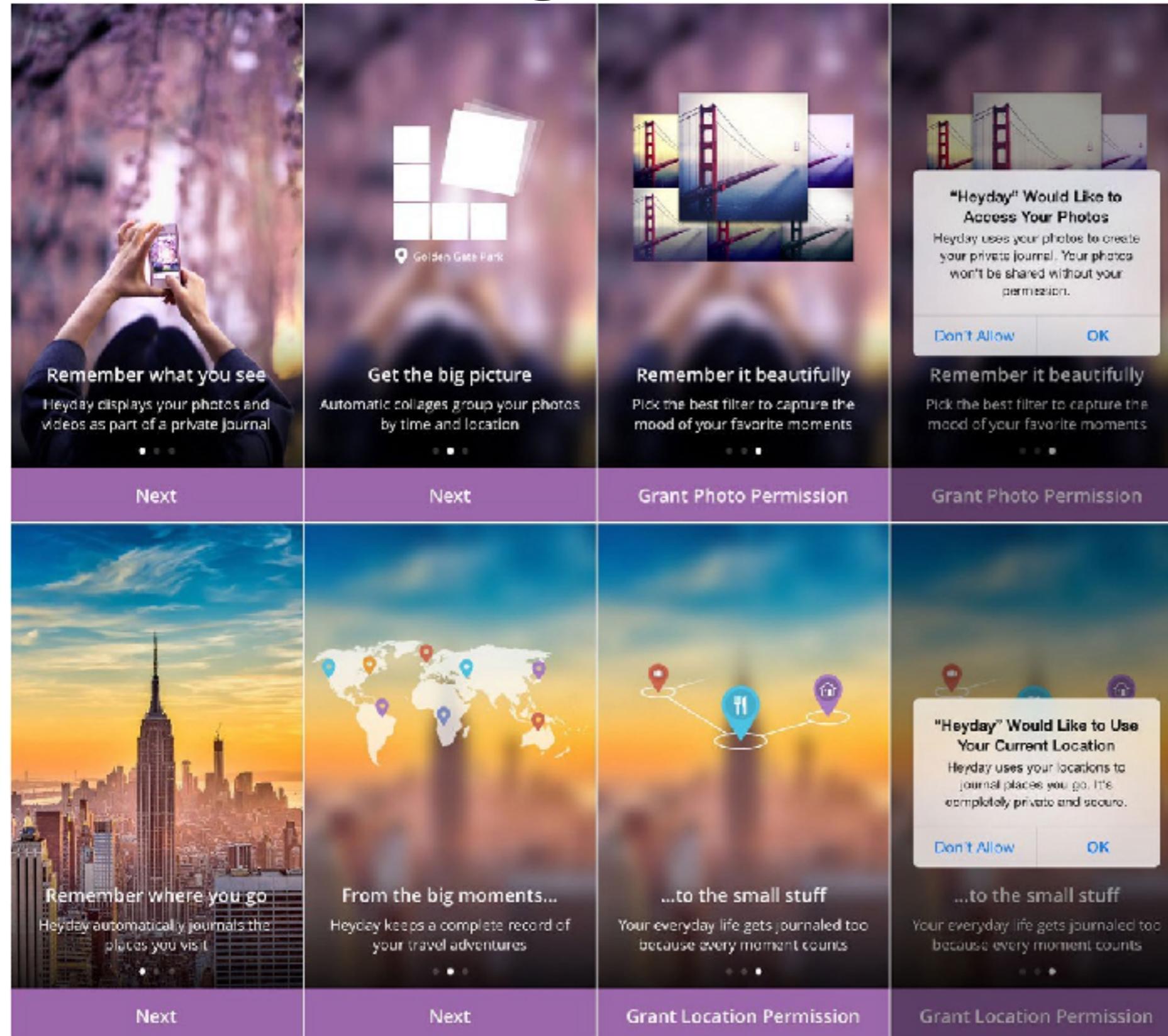
when asking for access



when asking for access



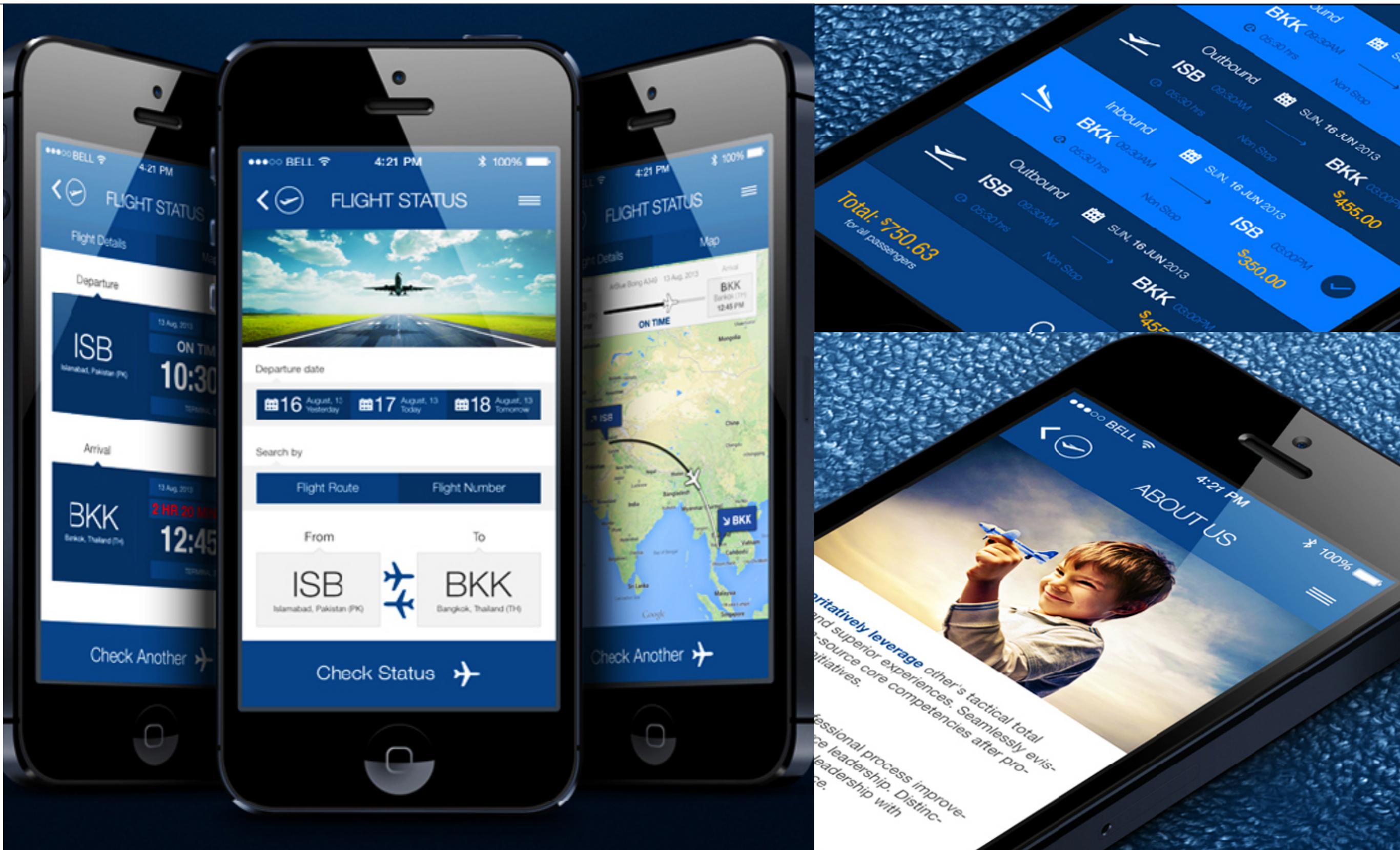
is this right?



what is good?

what is bad?

some great examples

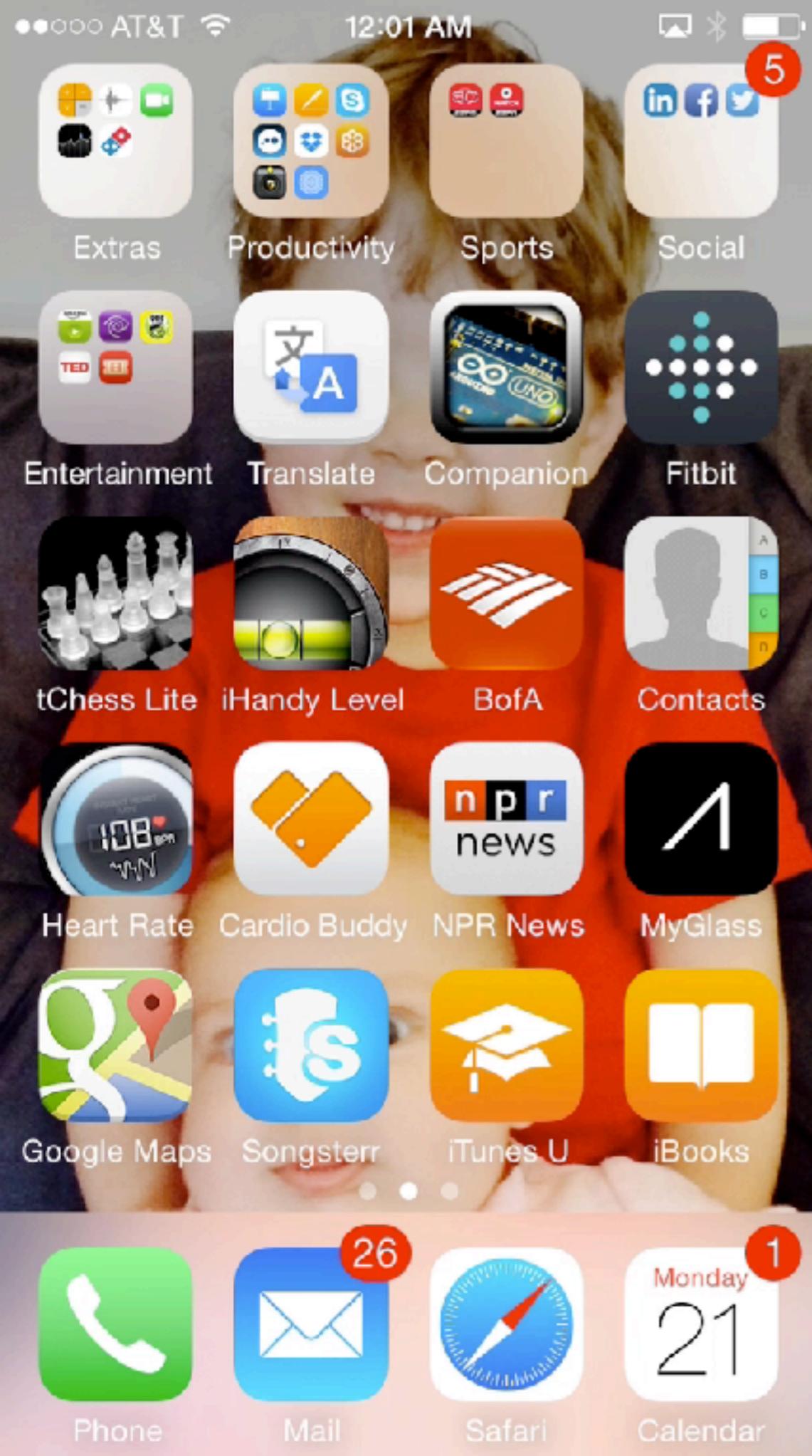


some great examples





an example

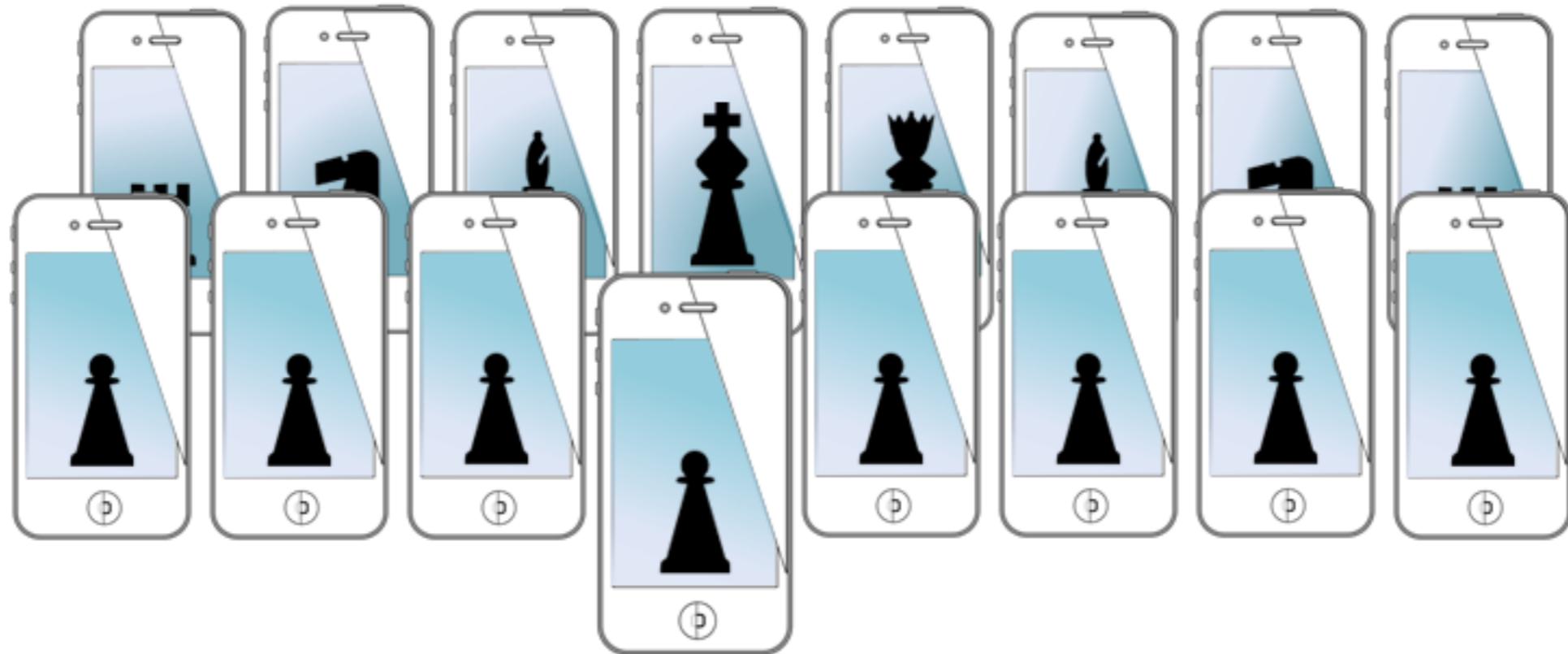


a **really** excellent
example

for next time...

- concurrency in iOS
 - blocks/closures
- introduction to audio sensing

MOBILE SENSING LEARNING

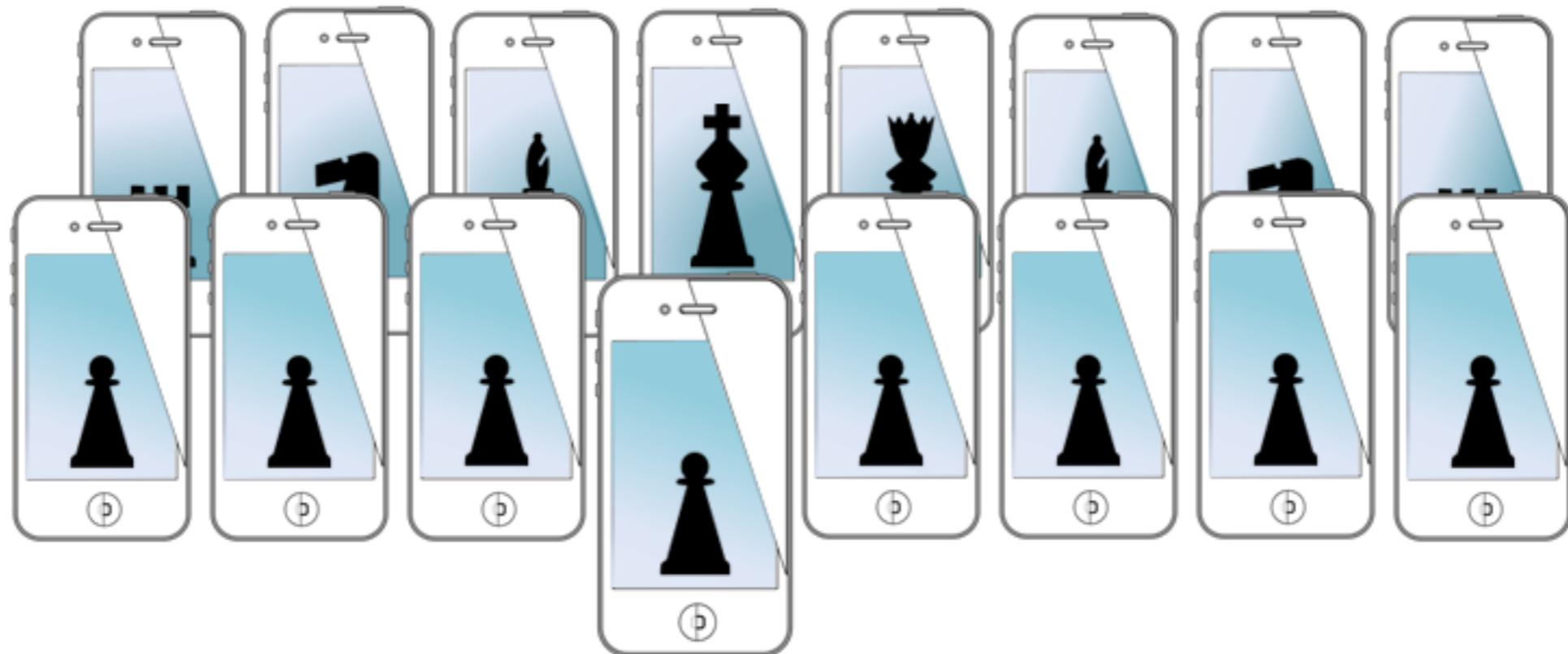


CS5323 & 7323
Mobile Sensing and Learning

week 3, lecture one: mobile design and interaction

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

MOBILE SENSING LEARNING



CS5323 & 7323
Mobile Sensing and Learning

queues, blocks, c++, audio session

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

agenda

- blocks and multi-threading
- objective c++ (no longer needed!)
- core audio intro

blocks

- not callback functions (but similar)
 - created at runtime
 - once created, can be called multiple times
 - can access data from scope when defined
 - syntax is `^(...)`
- not exactly a lambda (*but similar*)
 - but it acts like an object that can be passed as an argument or created on the fly
- also used in swift (called closures)

block syntax

Defining a block like a variable and setting “block code”:

return type

```
// create a block on the fly
float (^onTheFlyBlockThatAddsTwoInts)(int, int); // declare the block, try not to make unclear
// define the behavior of the block
onTheFlyBlockThatAddsTwoInts =^(int a, int b){
    return (float)(a+b);
}
// use the block
NSLog(@" On the fly value: %.4f",onTheFlyBlockThatAddsTwoInts(5,6));
```

block name

param types

define code that will execute

Using a block within a method call:

```
-----
//enumerate an Array with a block
NSArray *myArray = @[@34.5,@56.4567,@(M_PI)];

// here the block is created on the fly for the enumeration
[myArray enumerateObjectsUsingBlock:^(NSNumber *obj, NSUInteger idx, BOOL *stop) {
    // print the value of the NSNumber in a variety of ways
    NSLog(@"Float Value = %.2f, Int Value = %d",[obj floatValue],[obj integerValue]);
}];
```

enumerate with block

some semantics

- variables from same scope where block is defined are **read only**

```
NSNumber * valForBlock = @5.0;
```

- Unless you use keyword:

```
__block NSNumber * valForBlock = @5.0;
```

- classes hold a **strong** pointer to blocks they use
- blocks hold a **strong** pointer to __block variables
 - so using “self” would create a retain cycle

```
self.value = (some function in block)
```

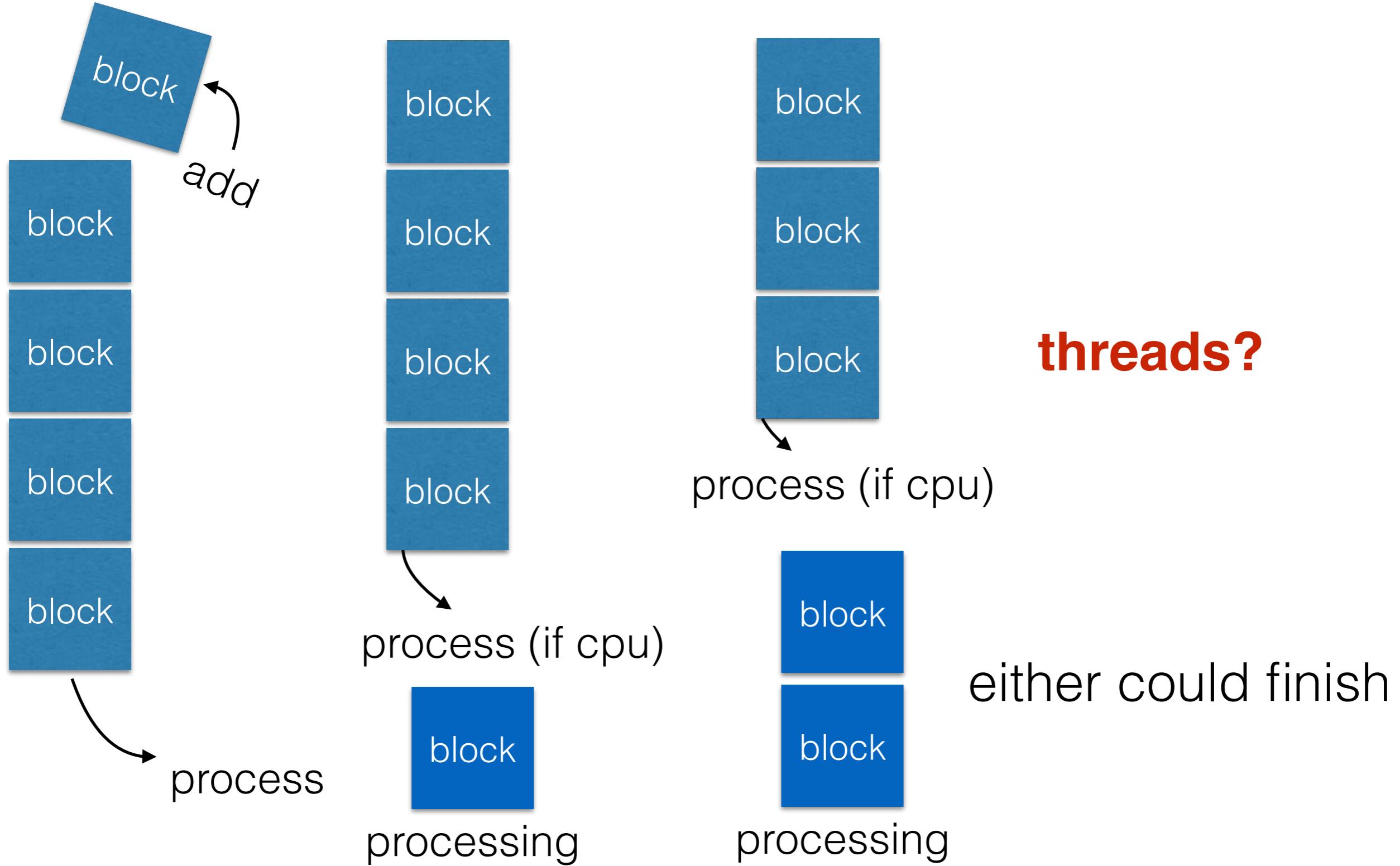
```
__block ViewController * __weak weakSelf = self;
```

```
weakSelf.value = (some function in block)
```

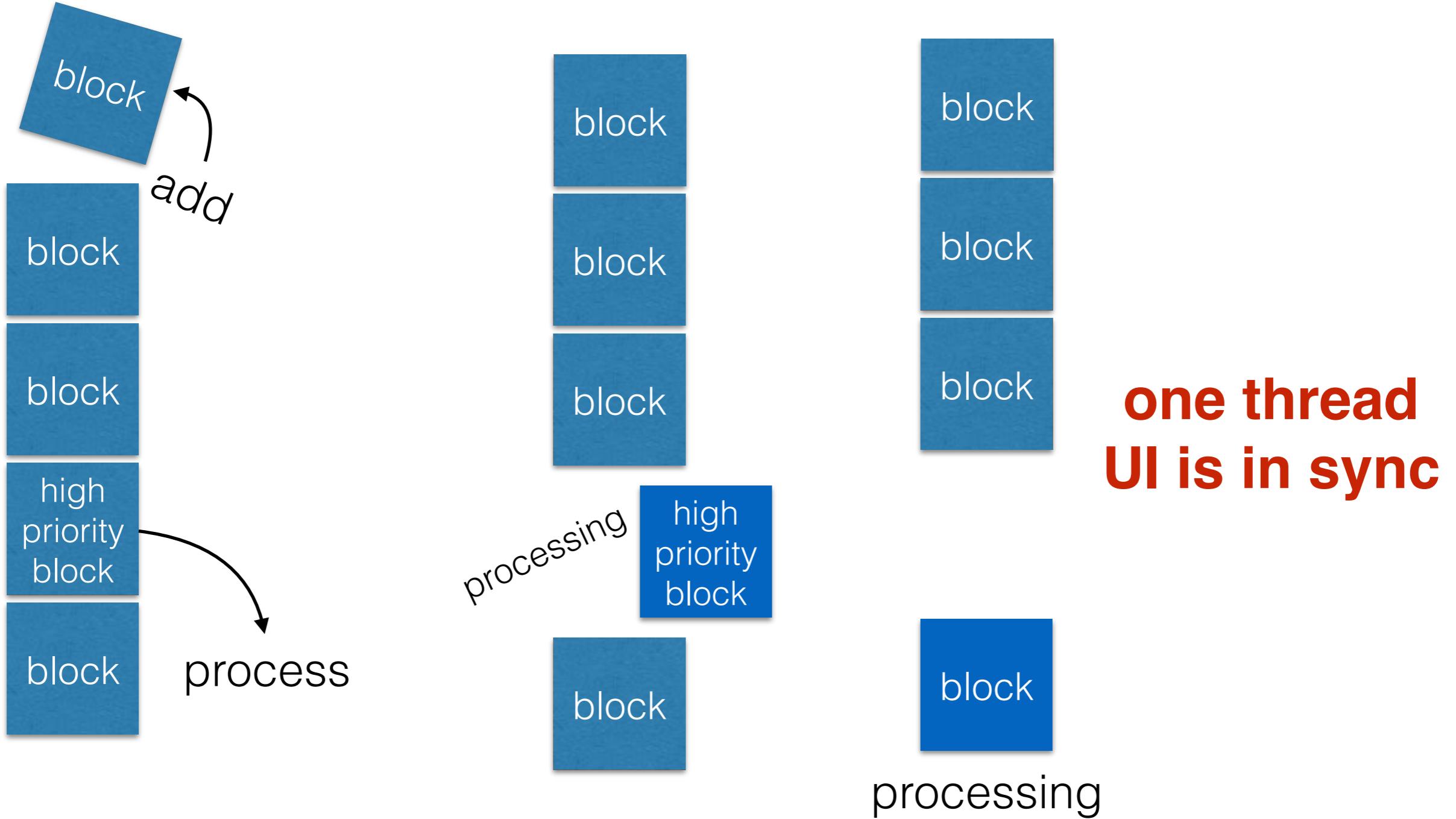
Concurrency in iOS

- grand central dispatch (GCD) handles all operations
 - GCD looks at “queues” of **blocks** that need to be run
 - GCD and the Xcode compiler work deep inside the OS, actually in the kernel – they are optimized
 - for a **serial queue** each block is run sequentially
 - for **concurrent queues** the first block is dequeued
 - if CPU is available, then the next block is also dequeued, but could finish any time
- the **main queue handles all UI operations** (and no other queue should generate UI changes!!)
 - so, **no updating of** the views, labels, buttons, (image views*) **except from the main queue**

concurrent queues



the main queue



queue syntax

```
// using c code:  
dispatch_queue_t someQueue = dispatch_queue_create("myCreatedQueue", DISPATCH_QUEUE_CONCURRENT);  
dispatch_async(someQueue, ^{  
    // your code to execute  
    for(int i=0;i<3;i++)  
        NSLog(@"I am being executed from a dispatched queue");  
  
    // now I need to set something in the UI, but I am not in the main thread:  
    // call from main thread  
    dispatch_async(dispatch_get_main_queue(), ^{  
        self.label.text = [NSString stringWithFormat:@"Finished running %d times, Safe",3];  
    });  
}); // this operation adds the block to the queue in a single clock cycle, then returns
```

```
NSOperationQueue *newQueue = [[NSOperationQueue alloc] init];  
newQueue.name = @"ObjCQueue";  
[newQueue addOperationWithBlock:^{  
    // your code to execute  
    for(int i=0;i<3;i++)  
        NSLog(@"I am being executed from a dispatched queue, from objective-c");  
  
    // now I need to set something in the UI, but I am not in the main thread!  
    // call from main thread  
    [self performSelectorOnMainThread:@selector(setMyLabel)  
        withObject:nil  
        waitUntilDone:NO];  
};
```

queue syntax

- using global queues

```
// An example of using already available queues from GCD
dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
    // your code to execute
    for(int i=0;i<3;i++)
        NSLog(@"I am being executed from a global concurrent queue");

    // now I need to set something in the UI, but I can't do that in the main thread!
    // call from main thread
    dispatch_async(dispatch_get_main_queue(), ^{
        self.label.text = @"Finished running from GCD global";
    });
});
```

access a global queue

not on main queue!!

main queue!

DISPATCH_QUEUE_PRIORITY_LOW

DISPATCH_QUEUE_PRIORITY_DEFAULT

DISPATCH_QUEUE_PRIORITY_HIGH

DISPATCH_QUEUE_PRIORITY_BACKGROUND

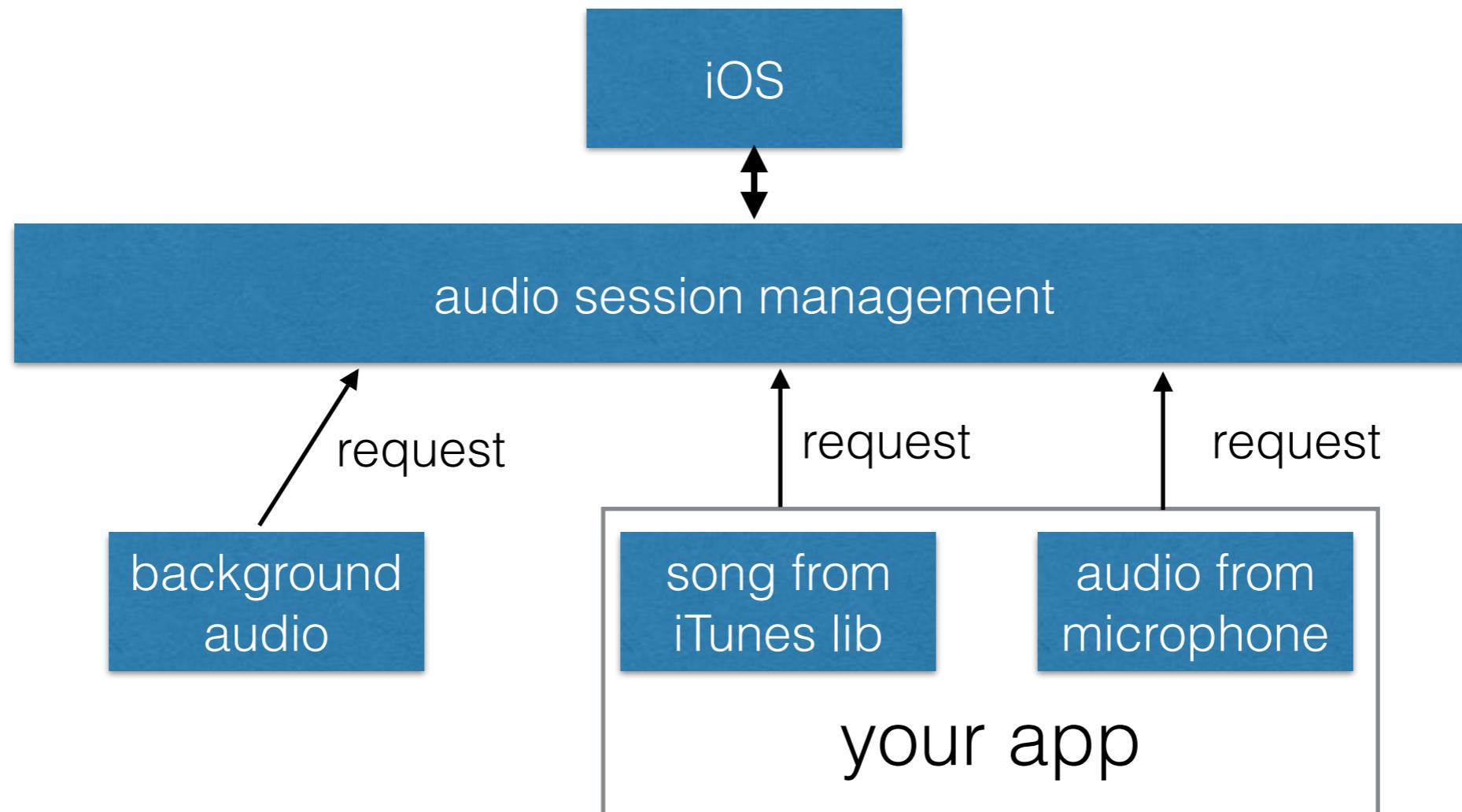
objective C++ (aside)

- this is no longer required for audio, but is good to know for later in the class when we use OpenCV...
- actually, its just c++
- ...but need to tell compiler we are using c++
- add any **#include** statements
- change **extensions to .mm** where you use c++ class(es)
- ARC won't help you for *malloc*, *calloc*, or **new**
 - so explicitly call *dealloc* and your class destructor
 - **delete** or *free*

Core Audio

- many **audio packages** exist, but we want **low level signals**
- **Audio Sessions** (high level, completely overhauled starting iOS7)
 - shared instance (for all applications)
 - set category (play, record, both)
 - choose options: like mixing with ambient sources
 - set audio route (new starting in iOS7)
 - set specific hardware within audio route
- **Audio Units** (more low level, output, input)
 - set stream format, buffer sizes, sampling rate,
 - initialize memory for audio buffers
 - set callback rendering procedure

audio sessions

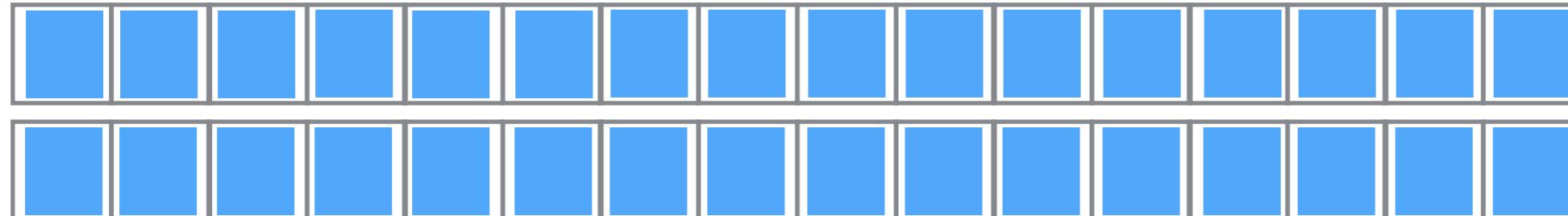


any request can alter management of other audio requests

mixing can be automatic — this is impressive

audio units

audio input buffer procedure, double buffer shown



(C)

sent to audio session callback

copy over samples, convert
exit from call as soon as possible!

do not allocate memory, take locks, or waste time!!

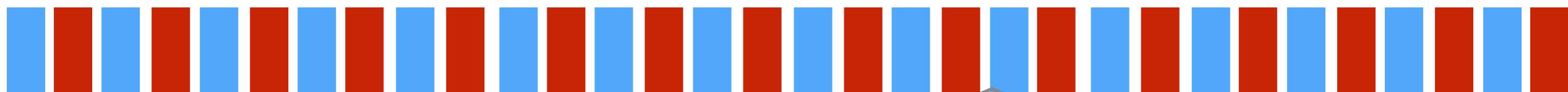


audio unit formats

microphone (input)



stereo could be interleaved (output)



right speaker

left speaker



callback preallocates buffers
developer fills the output buffer
OS handles playing the buffer
if you don't fill fast enough,
audio is choppy

wouldn't it be **great** if there was a module that **handled** all the specifics of **audio units for us?**

Novocaine: takes the pain out of audio processing

Originally developed by **Alex Wiltschko**

Heavily manipulated by **eclarson**



Alex Wiltschko
alexbw

 Twitter
 Boston, MA
 alex.bw@gmail.com
 Joined on Dec 4, 2009

audio made easy

- Novocaine (I mean real easy!!)

```
private lazy var audioManager:Novocaine? = {  
    return Novocaine.audioManager()  
}()  
private lazy var inputBuffer:CircularBuffer? = {  
    return CircularBuffer.init(  
        numChannels: Int64(self.audioManager!.numInputChannels),  
        andBufferSize: Int64(BUFFER_SIZE))  
}()
```

declare properties

setup manager and init buffer

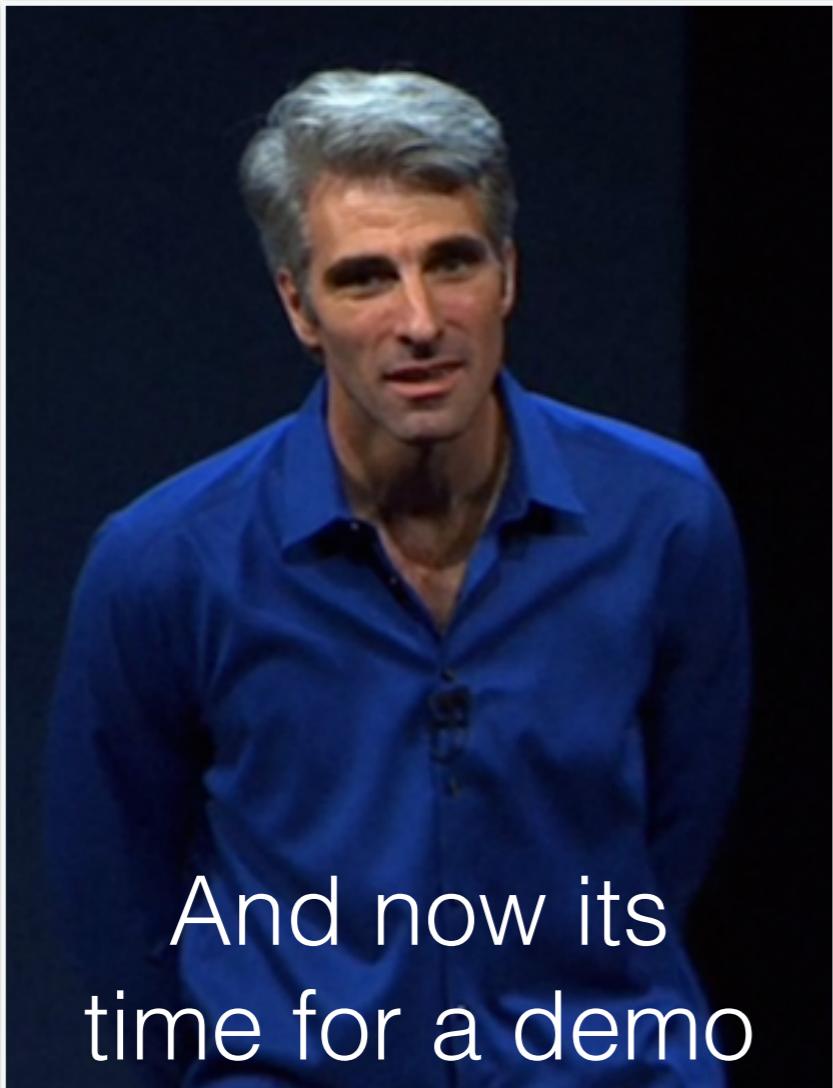
```
self.audioManager?.inputBlock = self.handleMicrophone  
  
private func handleMicrophone (data:Optional<UnsafeMutablePointer<Float>>,  
                           numFrames:UInt32,  
                           numChannels: UInt32) {  
    // copy samples from the microphone into circular buffer  
    self.inputBuffer?.addNewFloatData(data, withNumSamples: Int64(numFrames))  
}  
  
self.audioManager?.outputBlock = self.handleSpeakerQueryWithAudioFile  
  
private func handleSpeakerQueryWithAudioFile(data:Optional<UnsafeMutablePointer<Float>>,  
                                             numFrames:UInt32,  
                                             numChannels: UInt32){  
    self.outputBuffer?.fetchInterleavedData(data, withNumSamples:Int64(numFrames))  
}
```

microphone samples as float array

data to write to speakers

novocaine setup demo

source code on GitHub



And now its
time for a demo

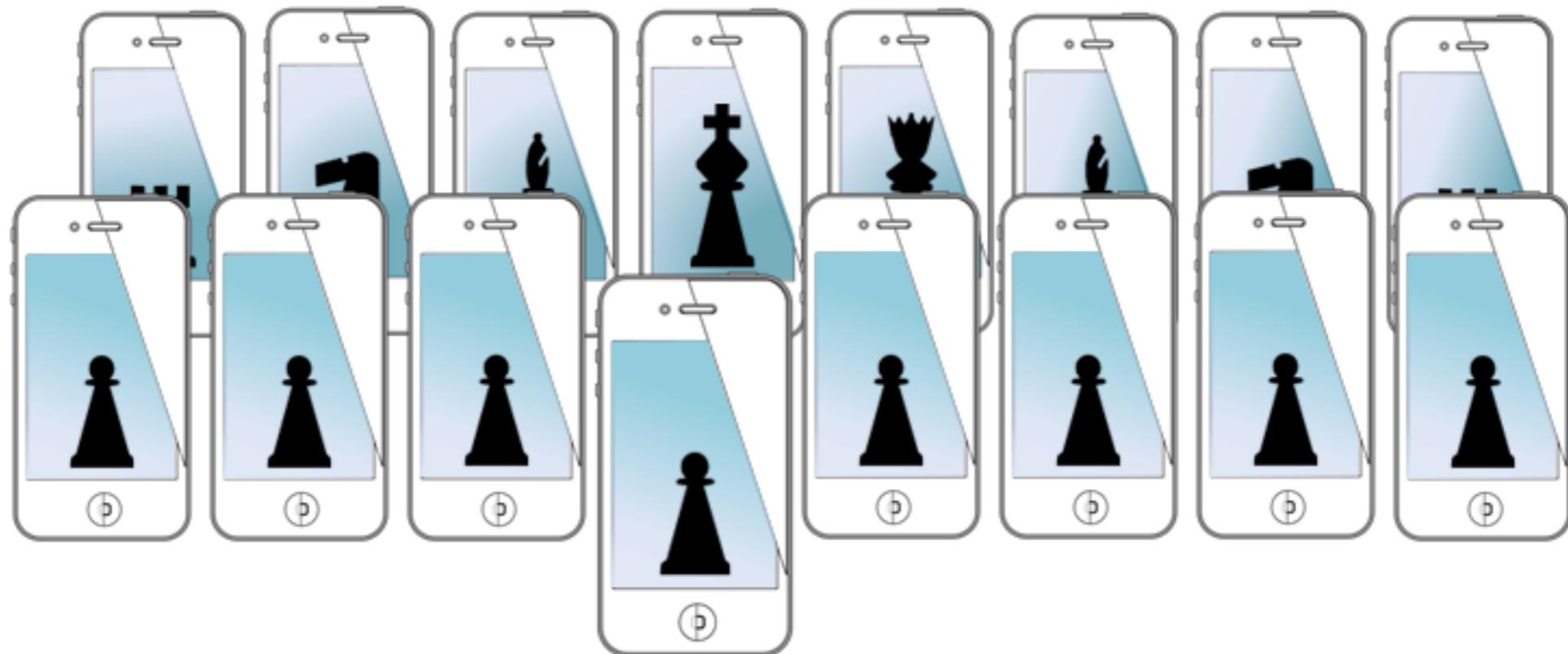


and rolling stones, if time

for next time...

- more core audio
 - playing songs (if not covered today)
 - getting samples from microphone
 - showing samples with Metal
 - working with sampled data
 - the accelerate framework

MOBILE SENSING LEARNING



CS5323 & 7323
Mobile Sensing and Learning

queues, blocks, c++, audio session

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