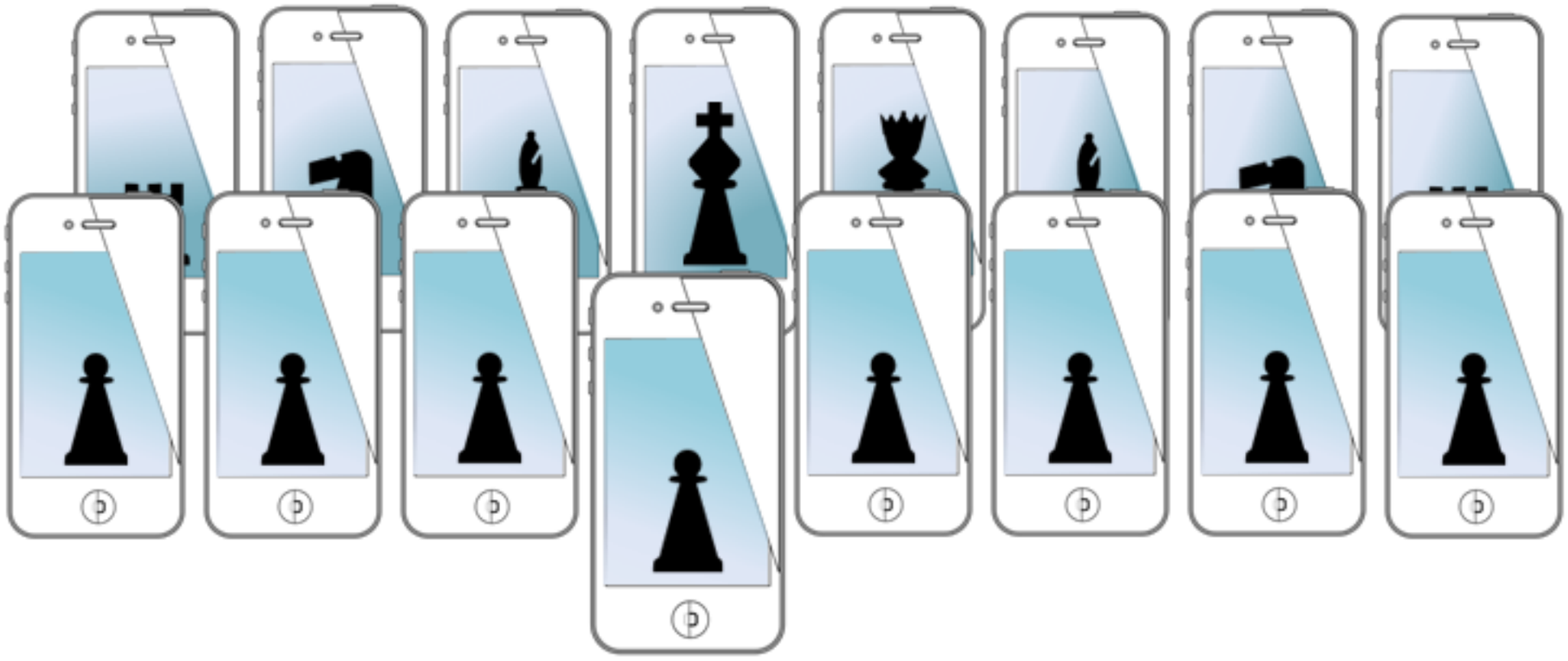
MOBILE SENSING LEARNING



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

clarity

text legible

adornment subtle

subtle visual motion cues

depth

planning

step one

look at core function

step one

look at core function

step one

look at core function

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

- **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

what do these users want?

who is your target audience?

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

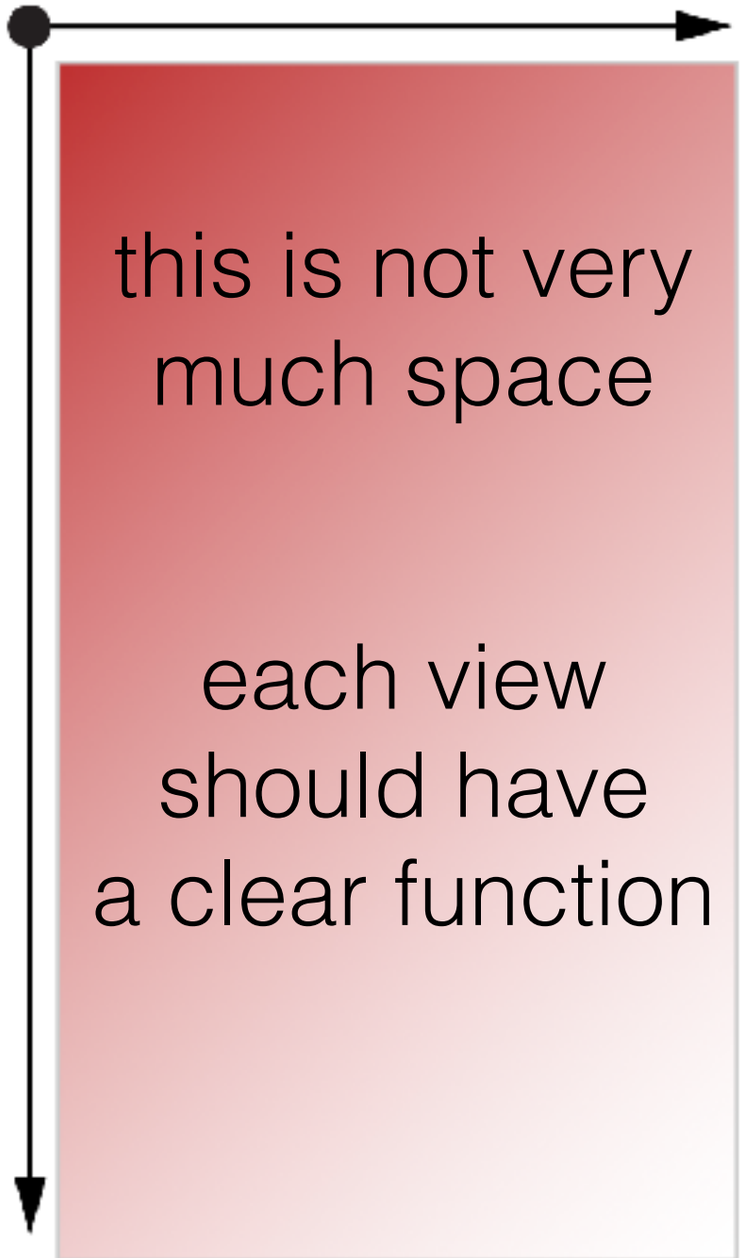
solution text

solution picker

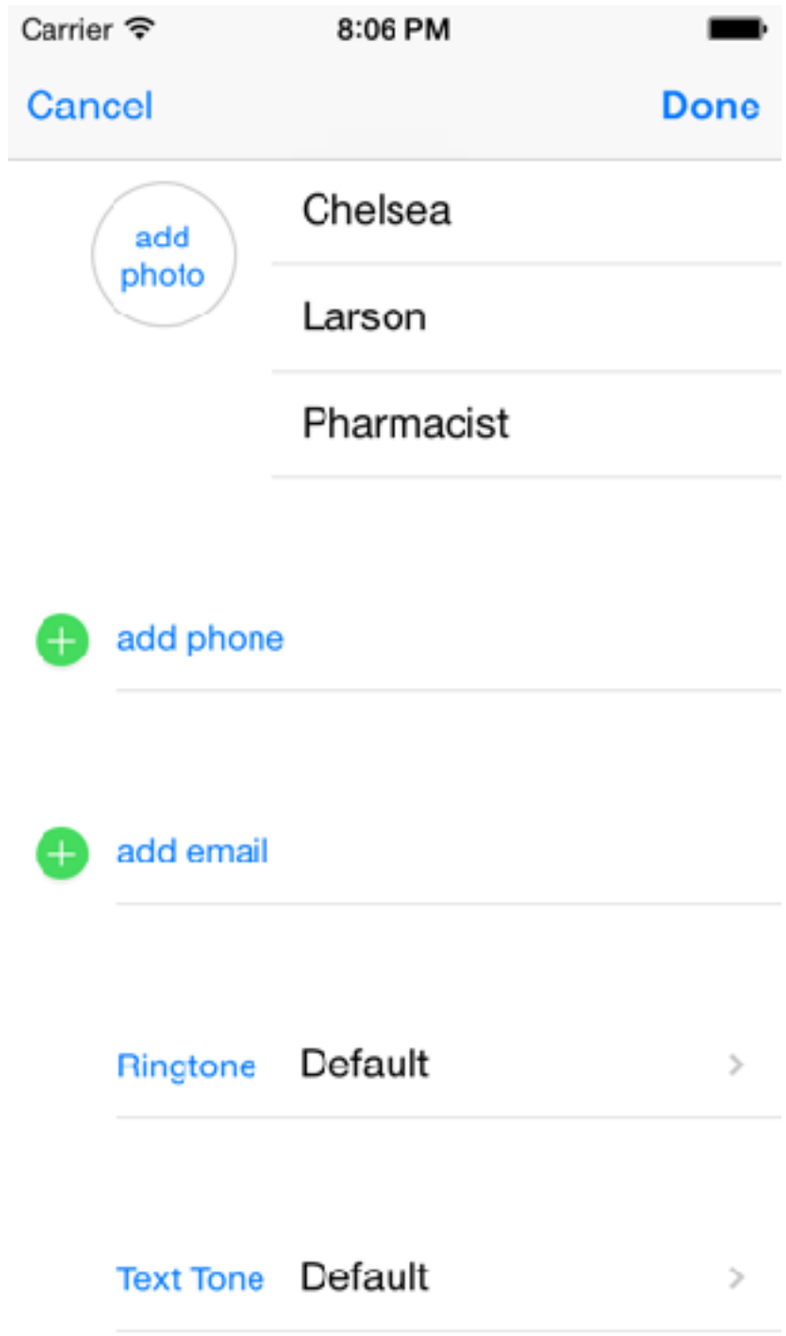
solution get from current location

layout with care

important



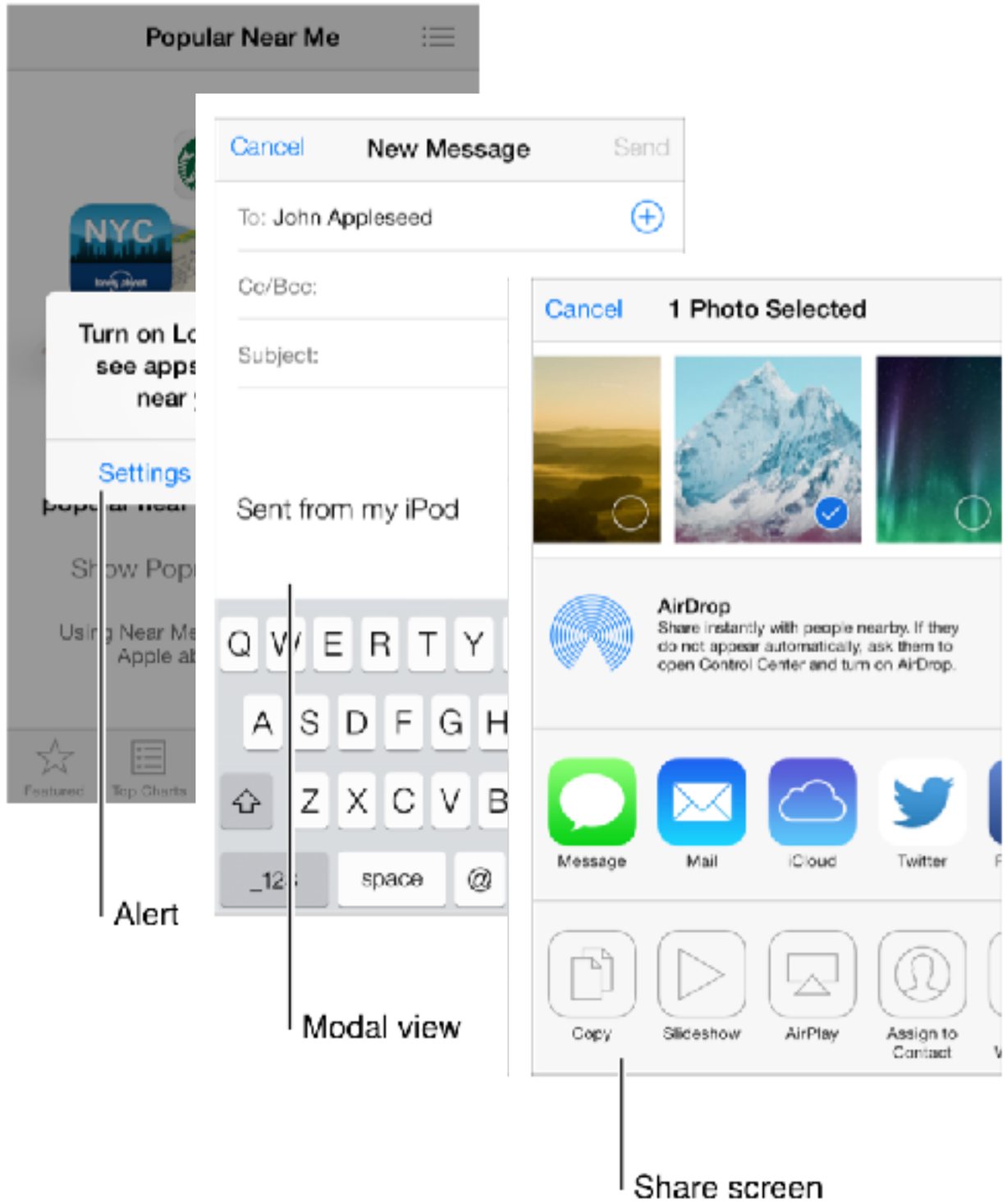
not so important



use modal sparingly

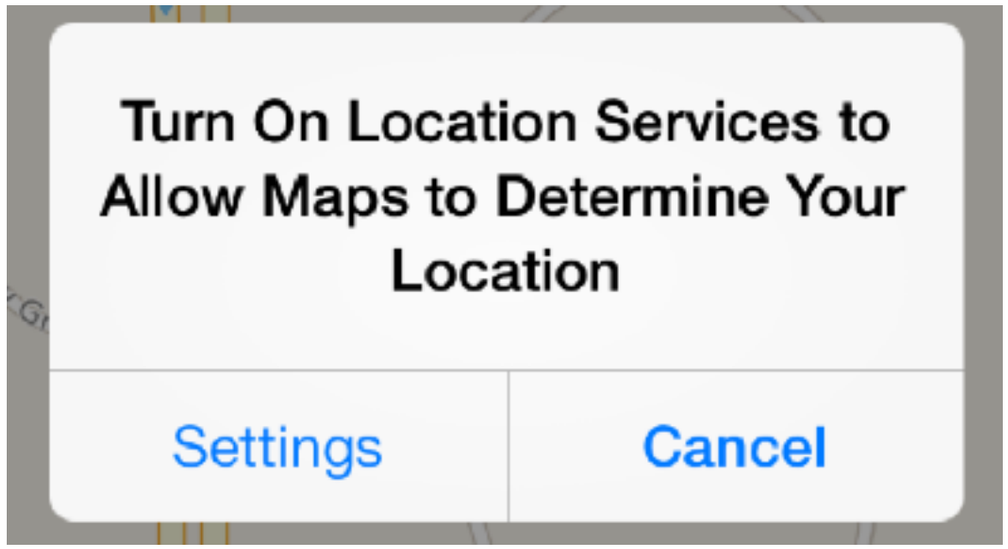
takes over the screen

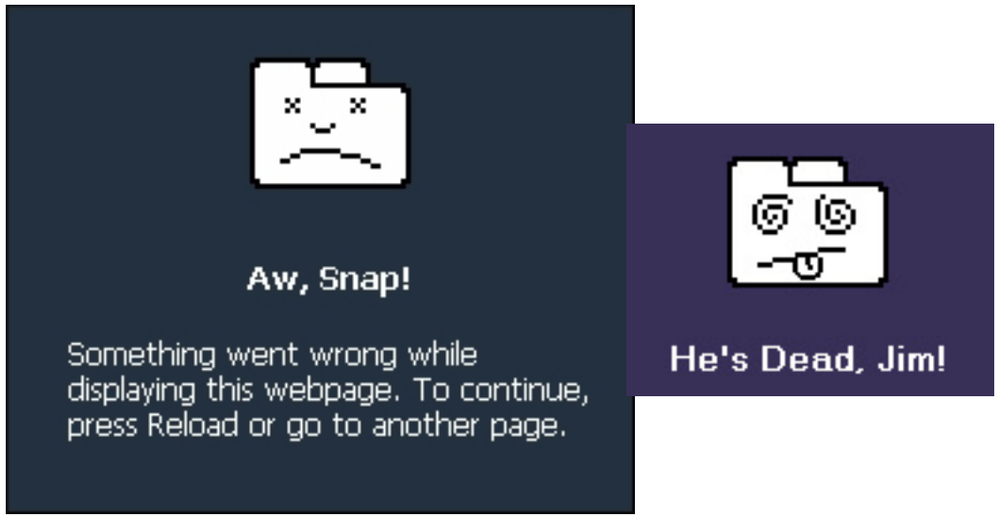
- prevents other interaction
- use when:
  - critical to capture attention
  - self contained action
    - like sending email
- keep simple, not tied to nav
- provide easy, safe exit



keep alerts succinct

- change wording to be clear and concise
- avoid jargon





Problem Connecting

There might be a problem with the server. Do you want to notify our staff about the issue?

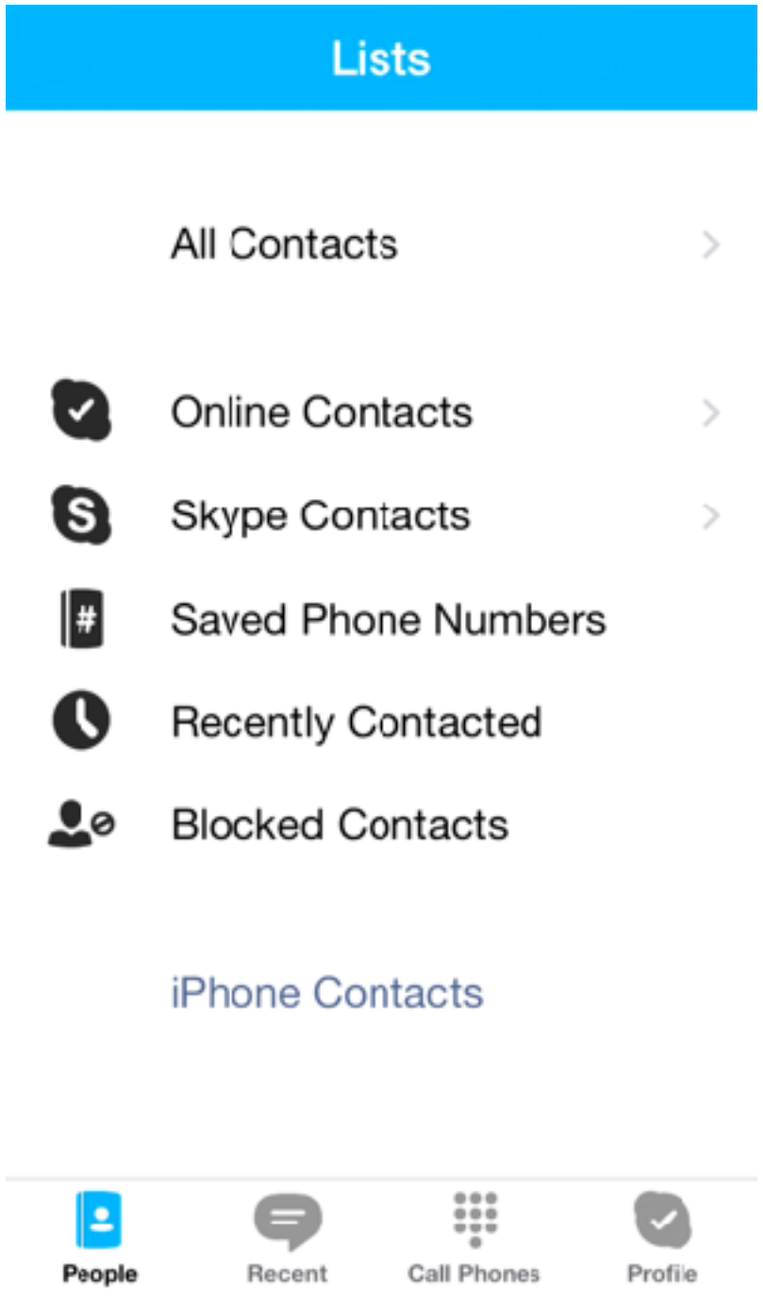
Send Notification | **Go Back**

- 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

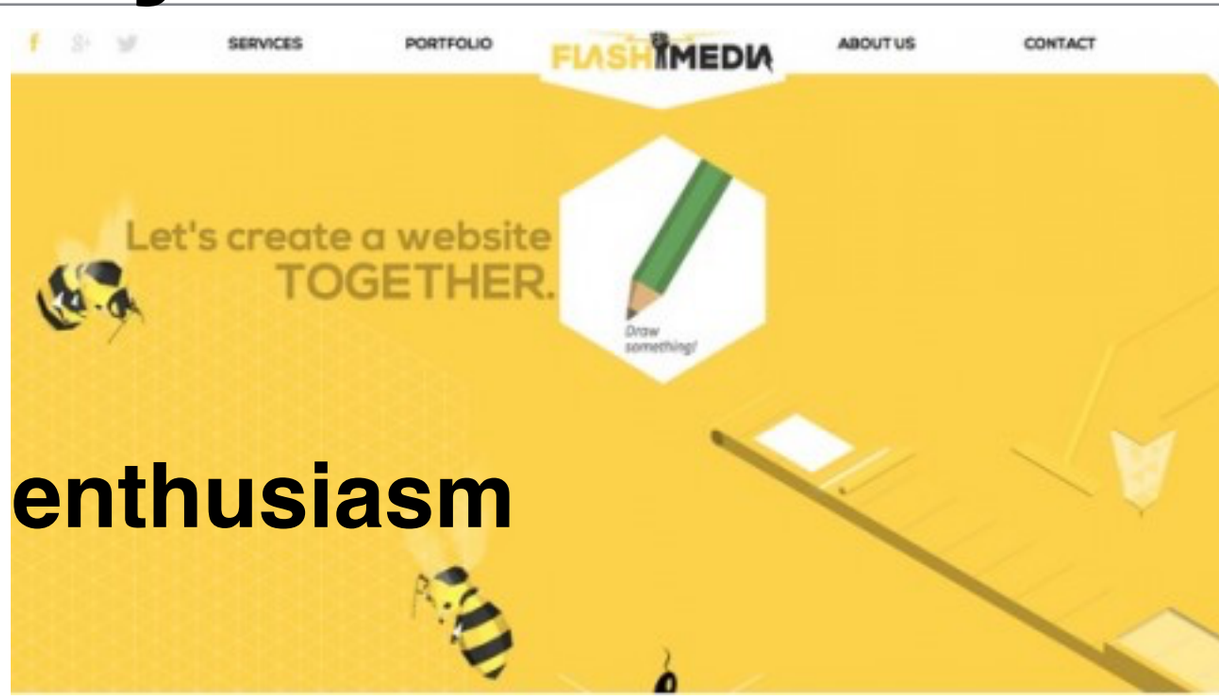
- use the splash screen to give impression of quickness

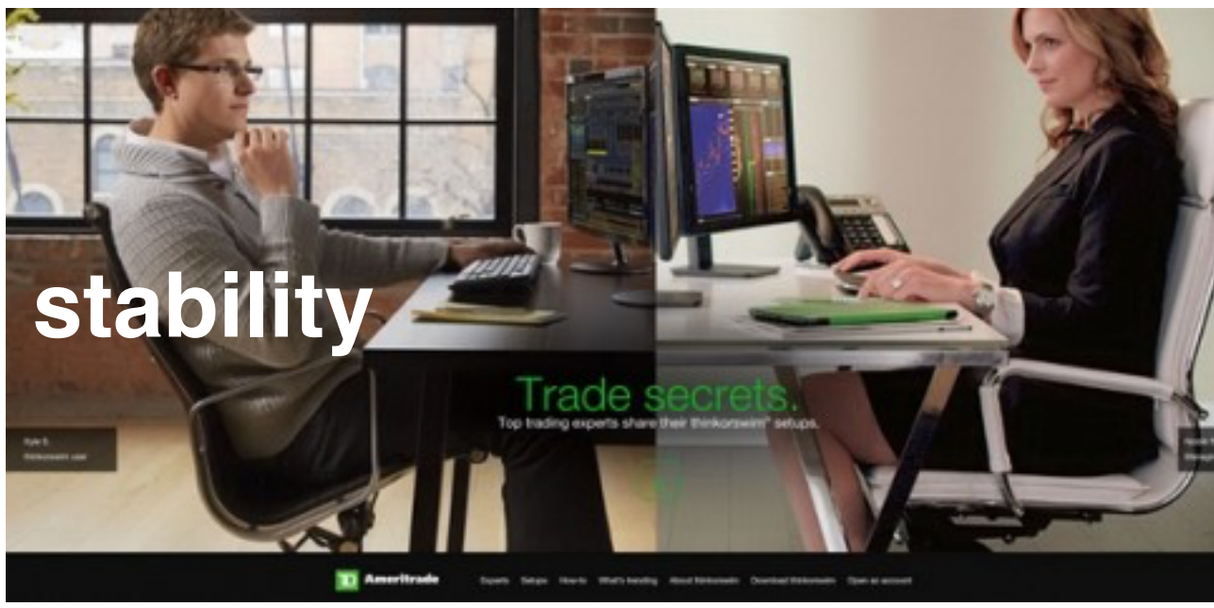
no



yes

use color “theory”

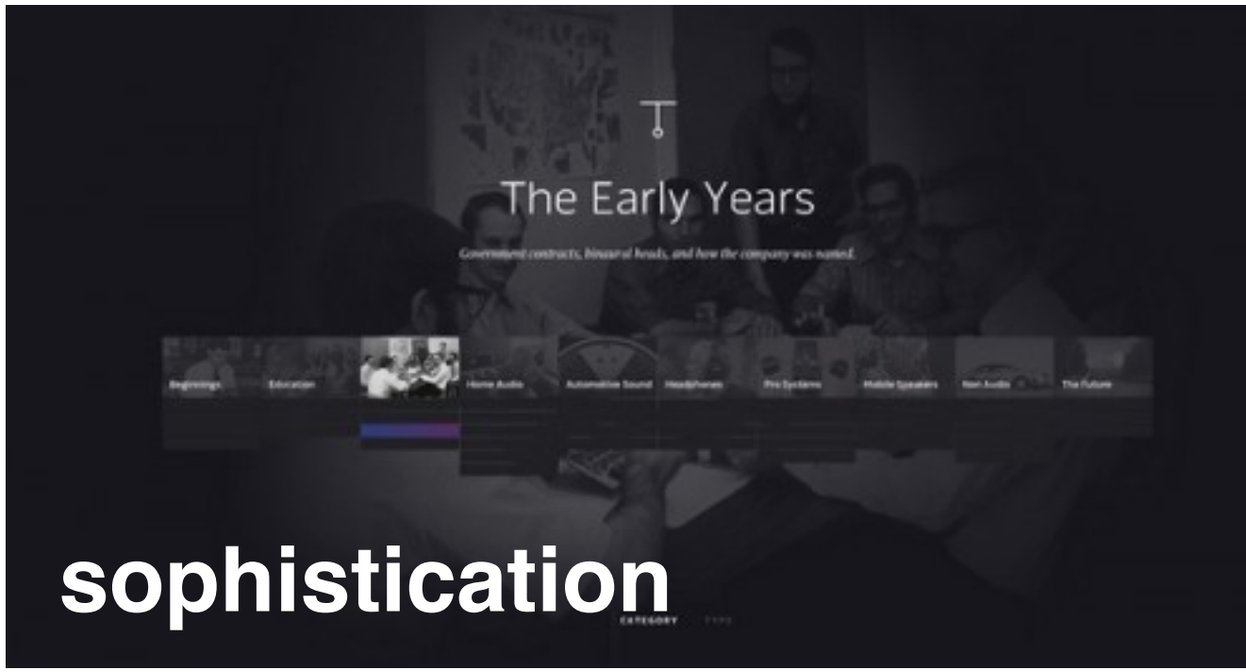




http://thenextweb.com/dd/2015/04/07/how-to-create-the-right-emotions-with-color-in-web-design/

use color “theory”

calm



sophistication

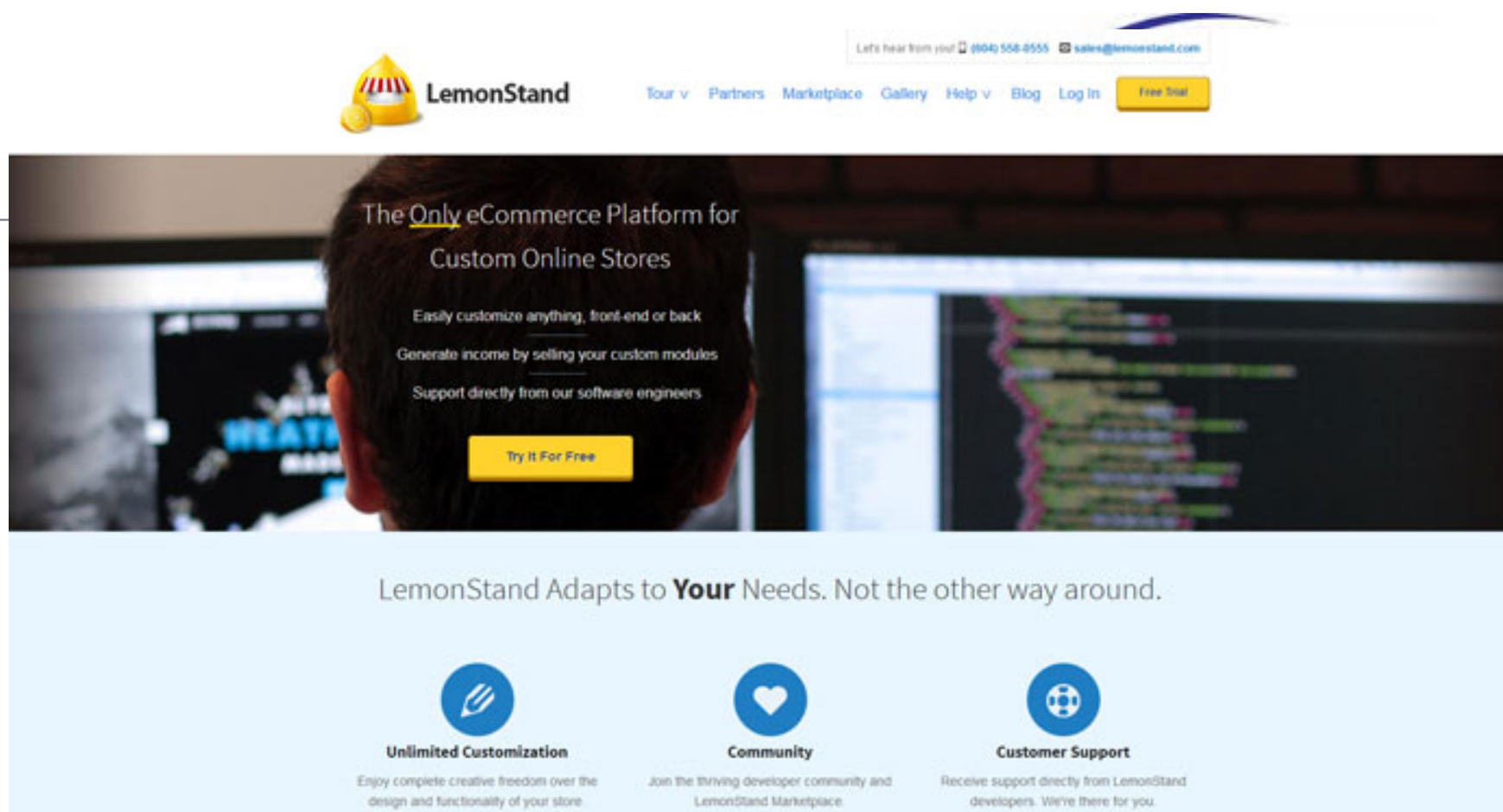
simplicity

luxury

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

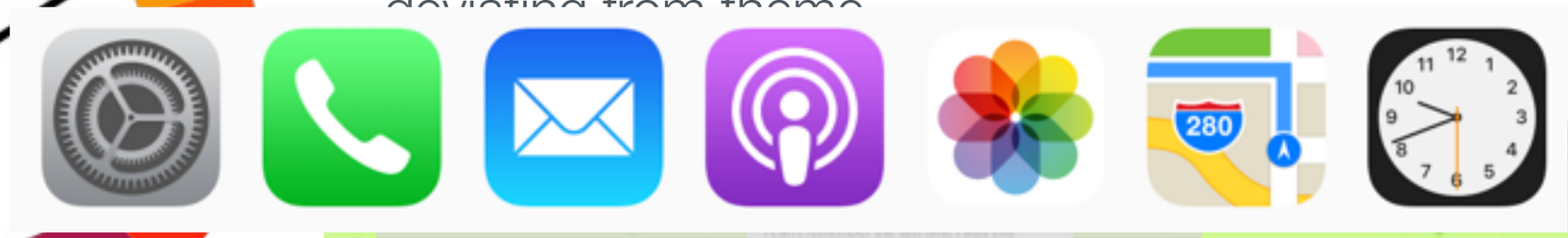
use colo

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

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

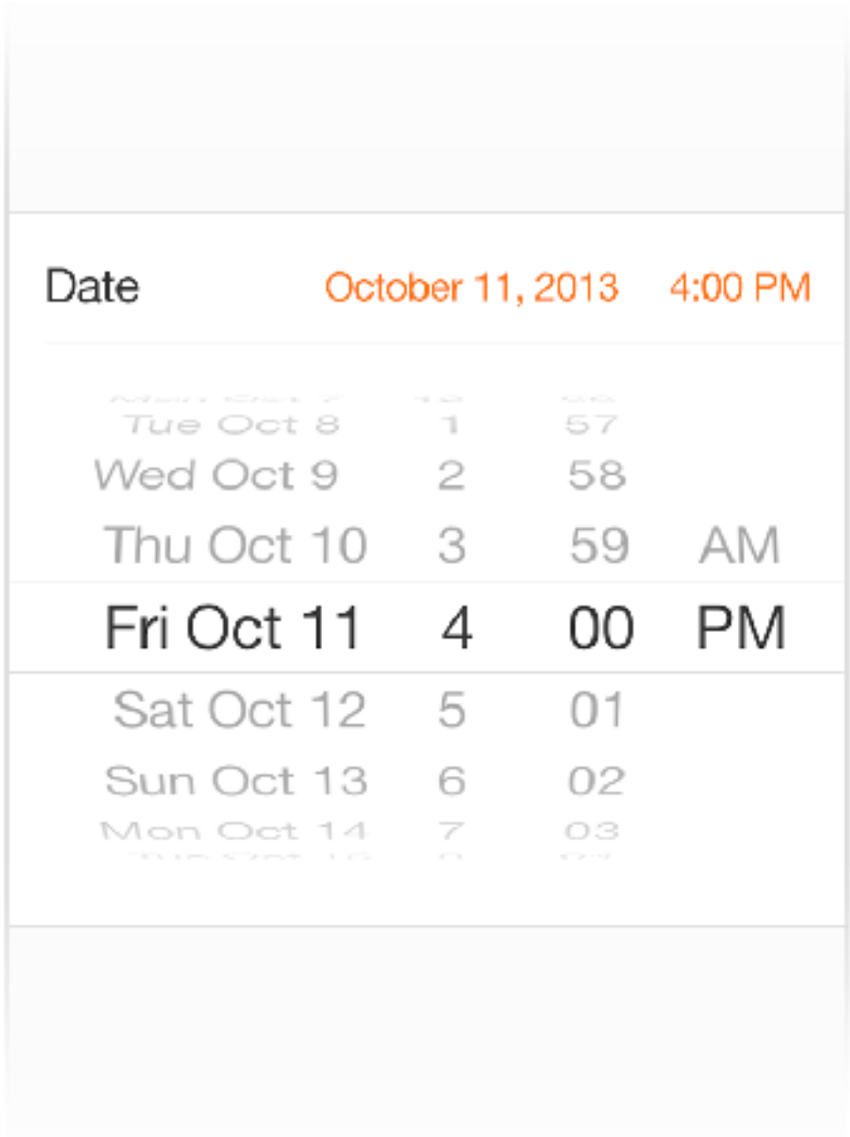
Apple says… design is…

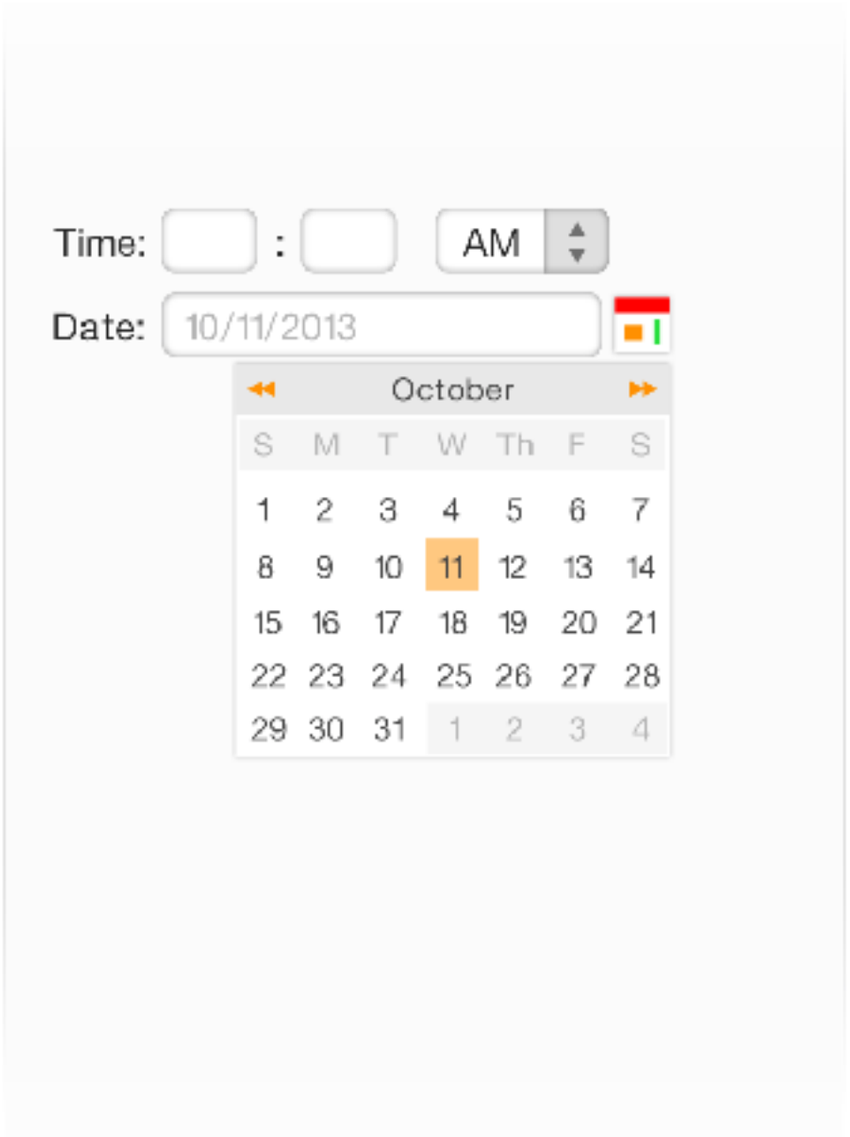
formatting


9:41 AM
100%
The Roasting
The Coffee App has everything from cultivating the coffee beans to serving it up in your cup! Explore the process it takes to make coffee from start to finish.

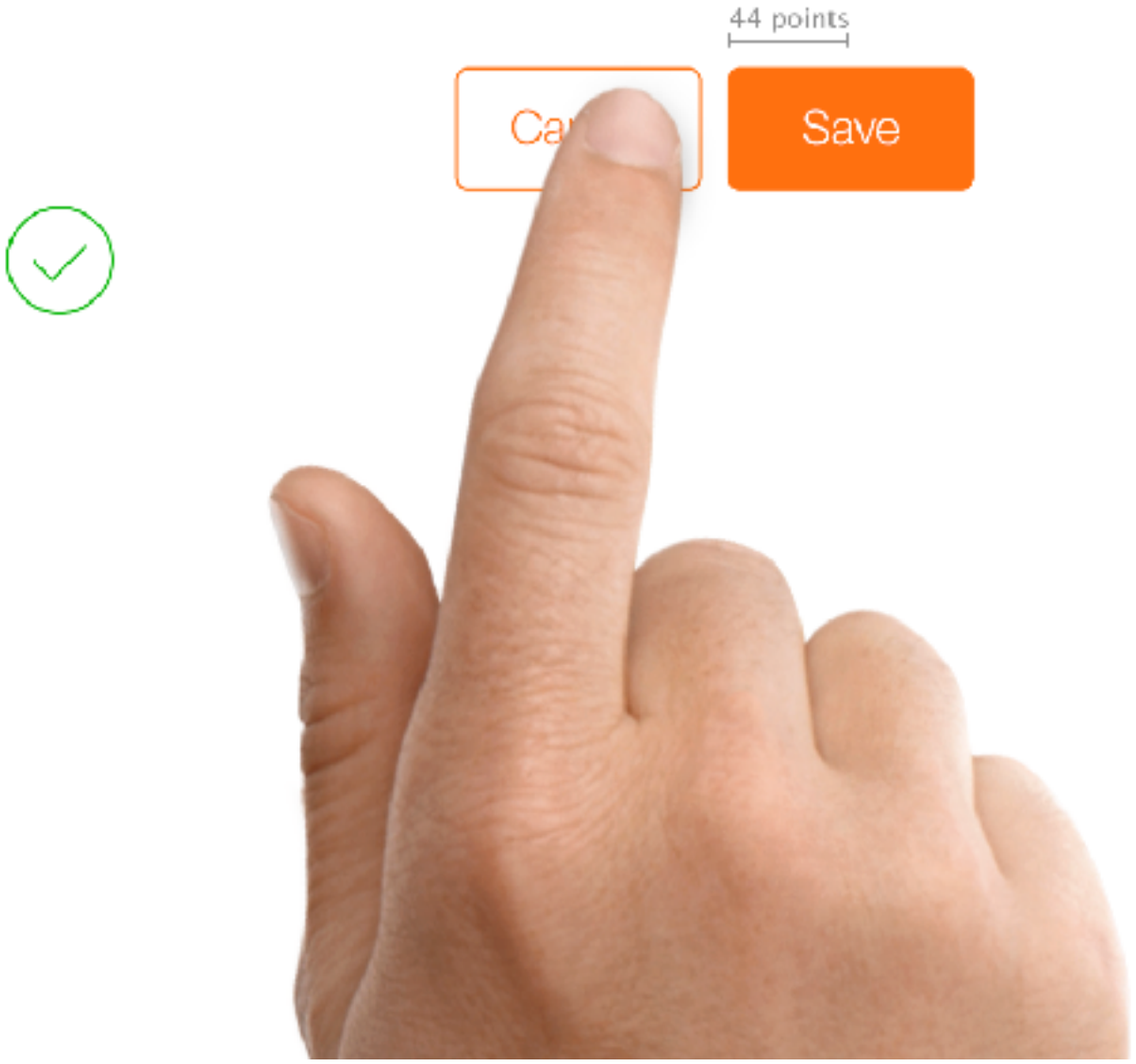

9:41 AM
100%
The Roast
The Coffee App has
from cultivating the
to serving it up in yo
Explore the process
make coffee from st
Learn Mo
Ways to Prepare
The Coffee App h
everything from c
coffee beans to s
in your cup! Expl

design for touch





design for taps



legible text

Headling

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.

Headling
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.

high contrast

Headling

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.

Headling

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

Headling

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.

Headling

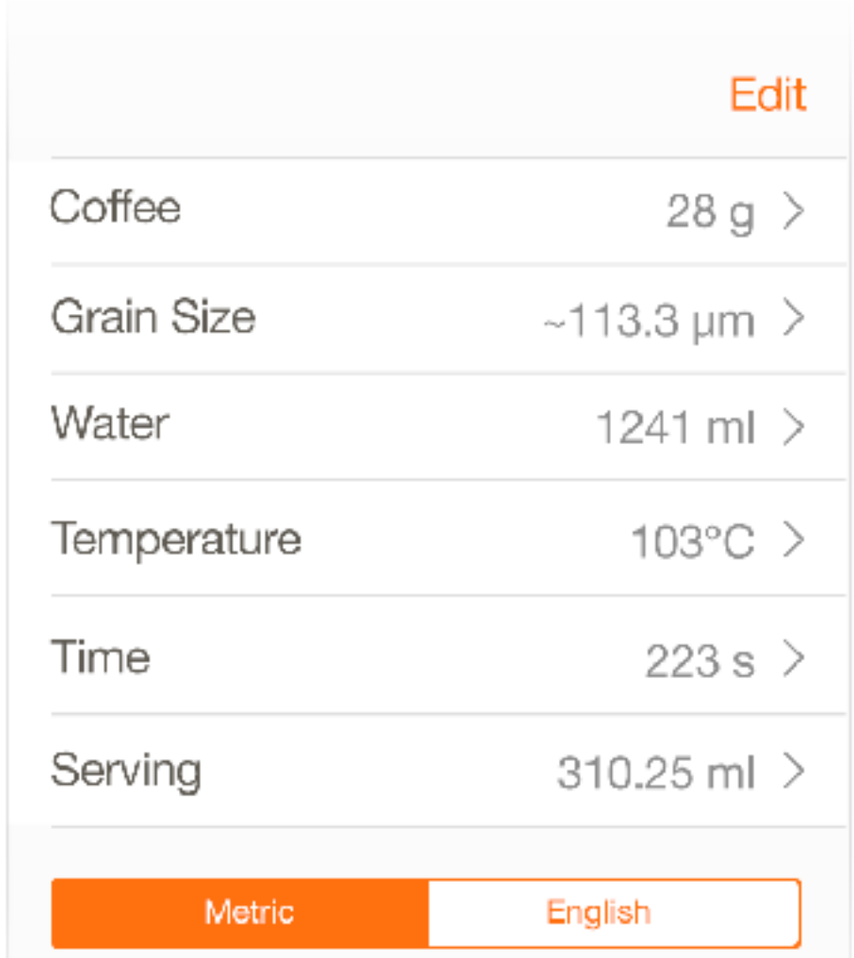
Sub-Headline

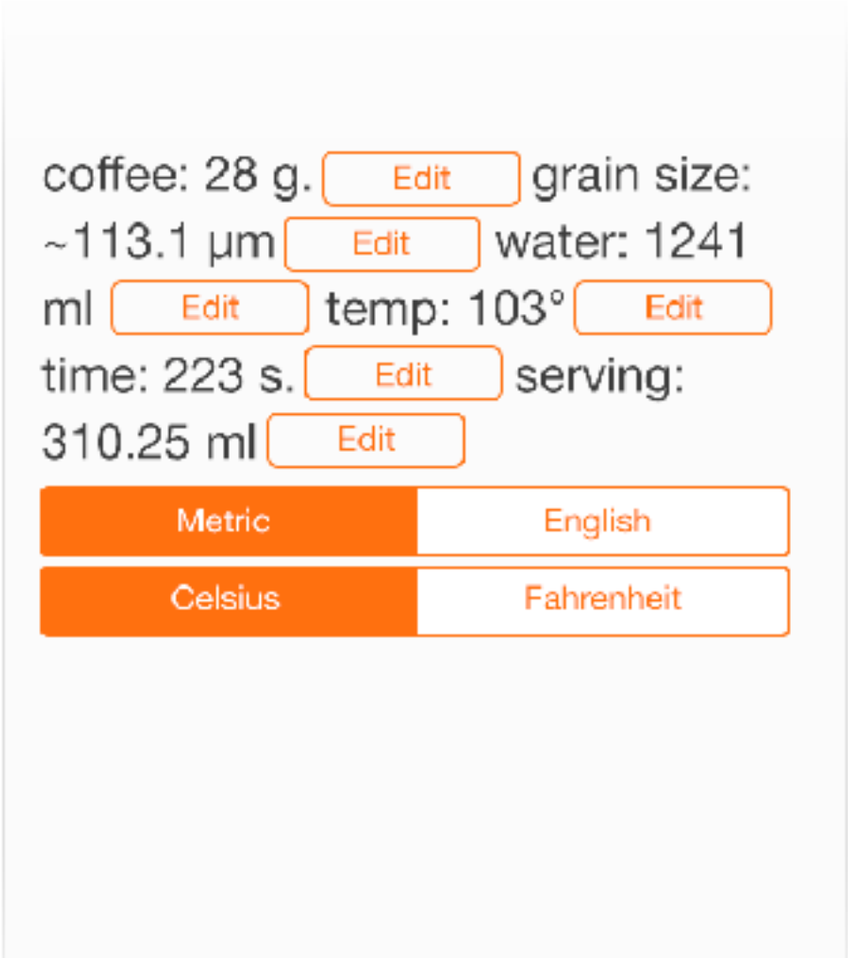
Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend egestas nisl vehicula nec. Nullam varius est dui, nec accumsan lectus posuere ut. Nullam viverra purus laoreet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

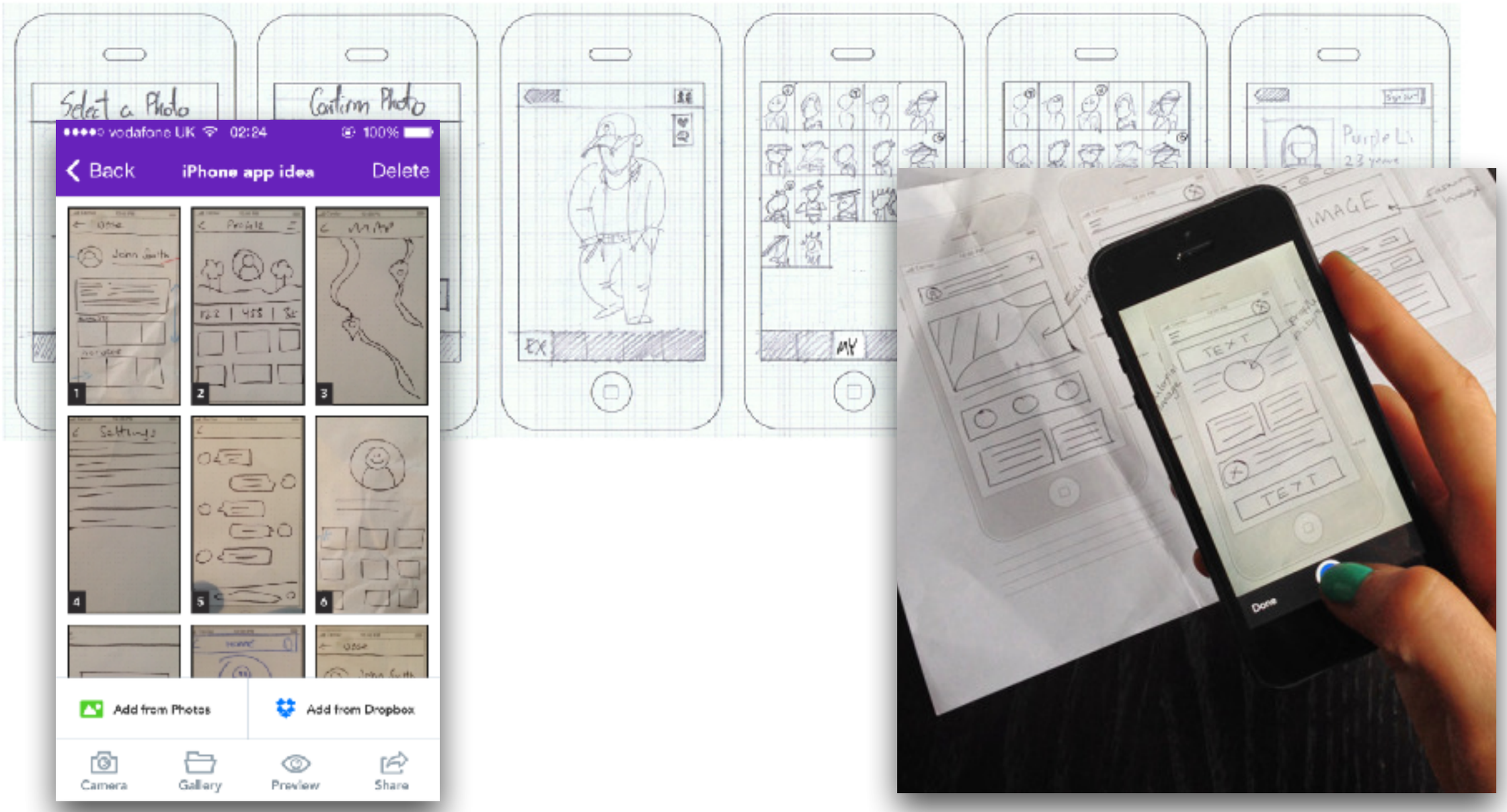
**ornare mperdiet blandit lectus. Morbi tristique*

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

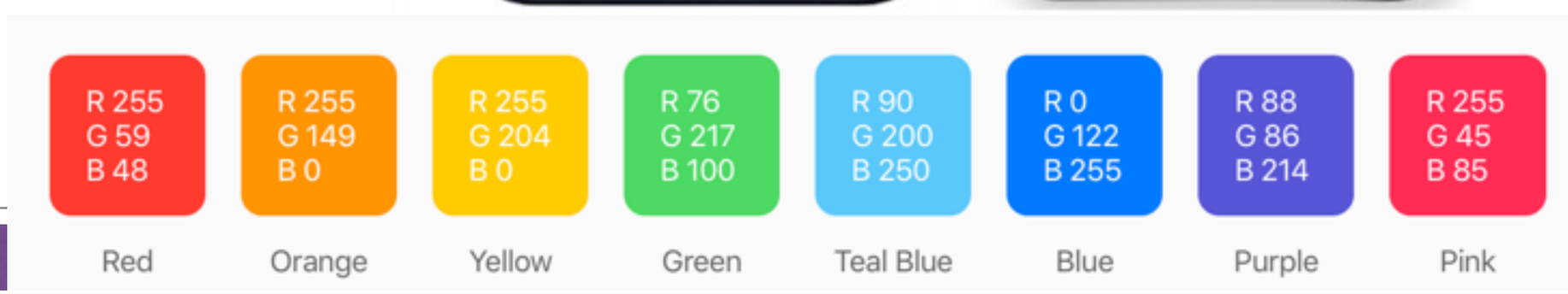
- no need for visual metaphor if intuitive
- make minimal and efficient
- color is king

flat design

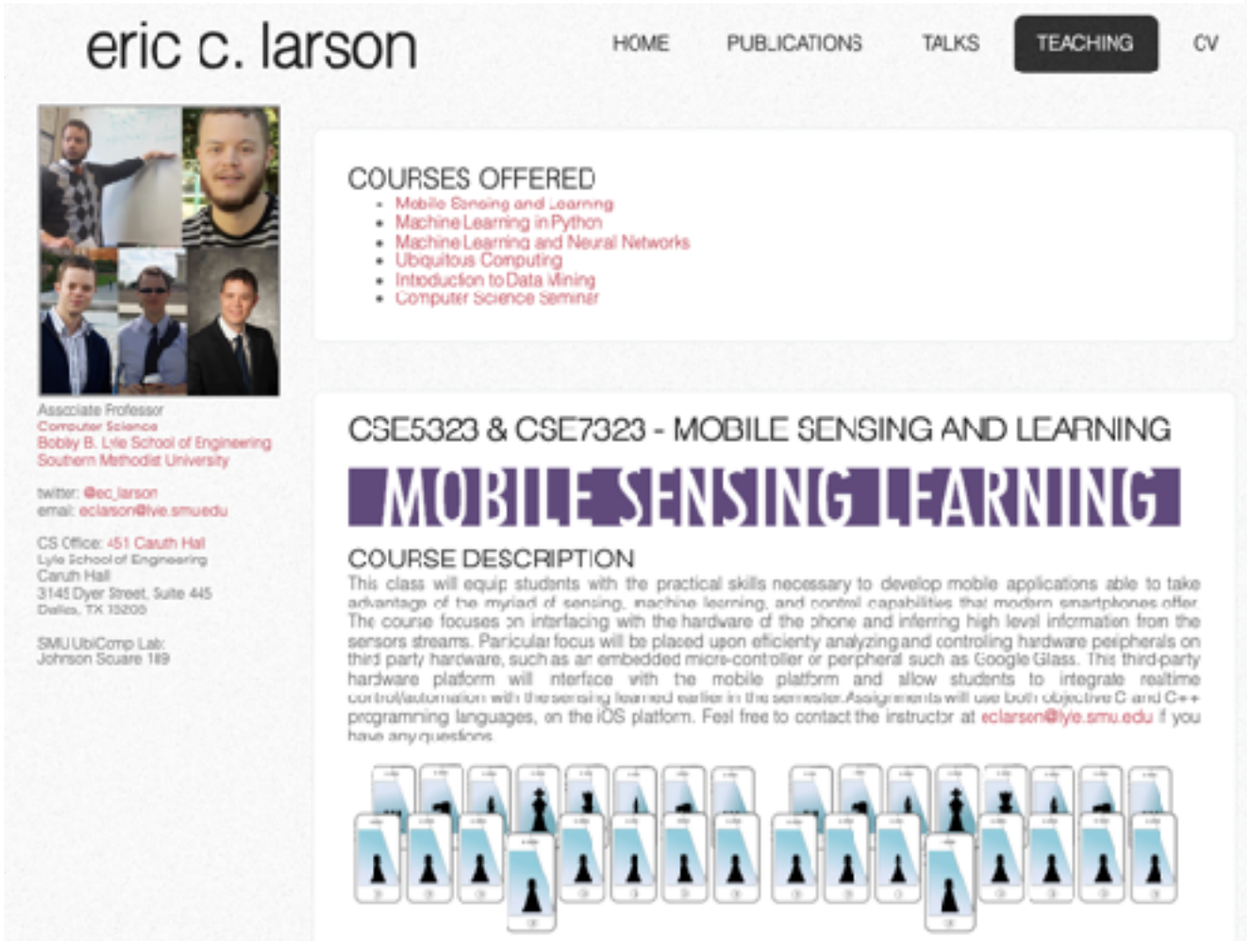
- main principles started in advertising
- the swiss style and minimalist style

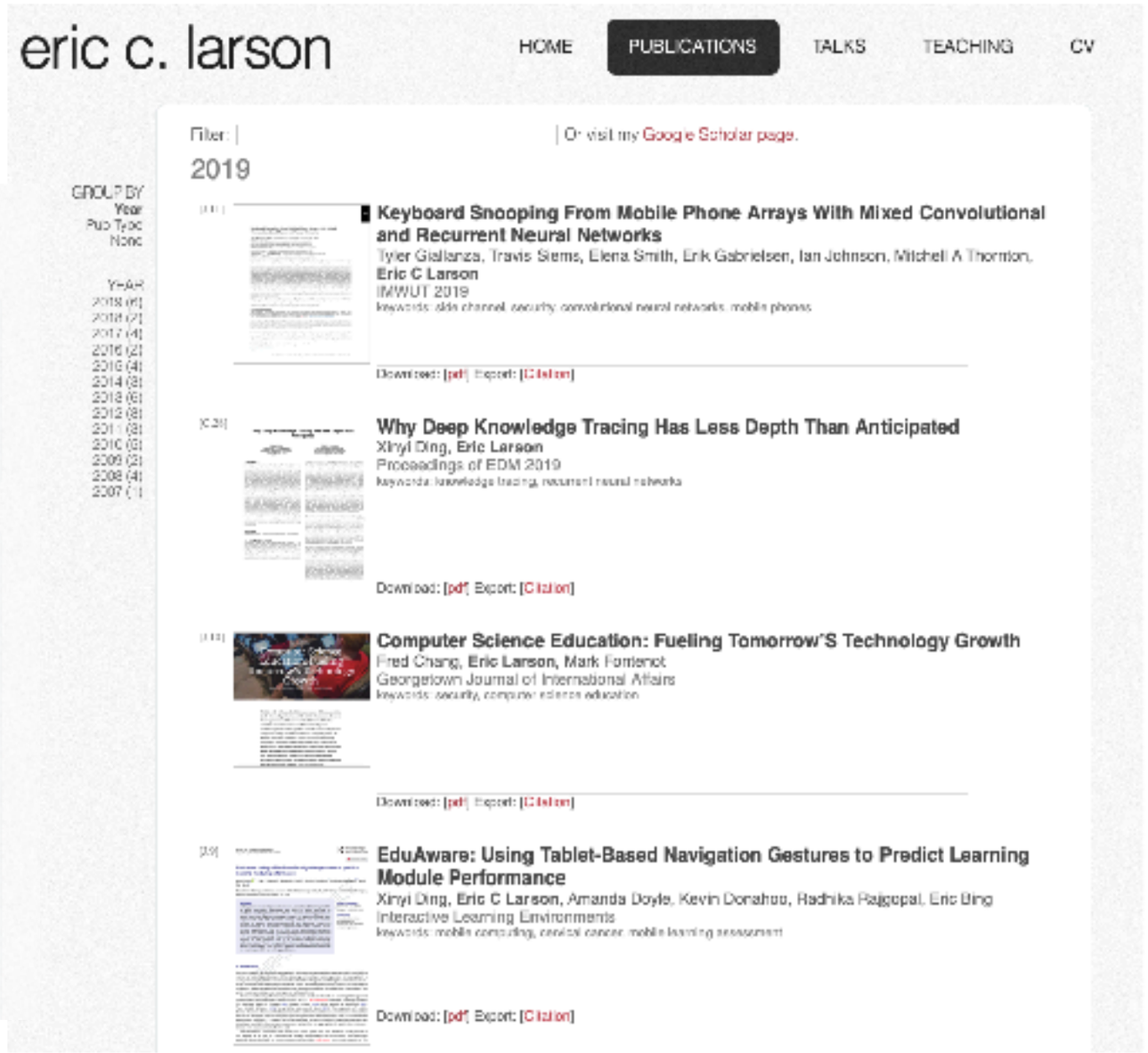
pure flat design

- 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



pure flat design?





Upgrade to Google+

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.

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.

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.

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.

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.

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.

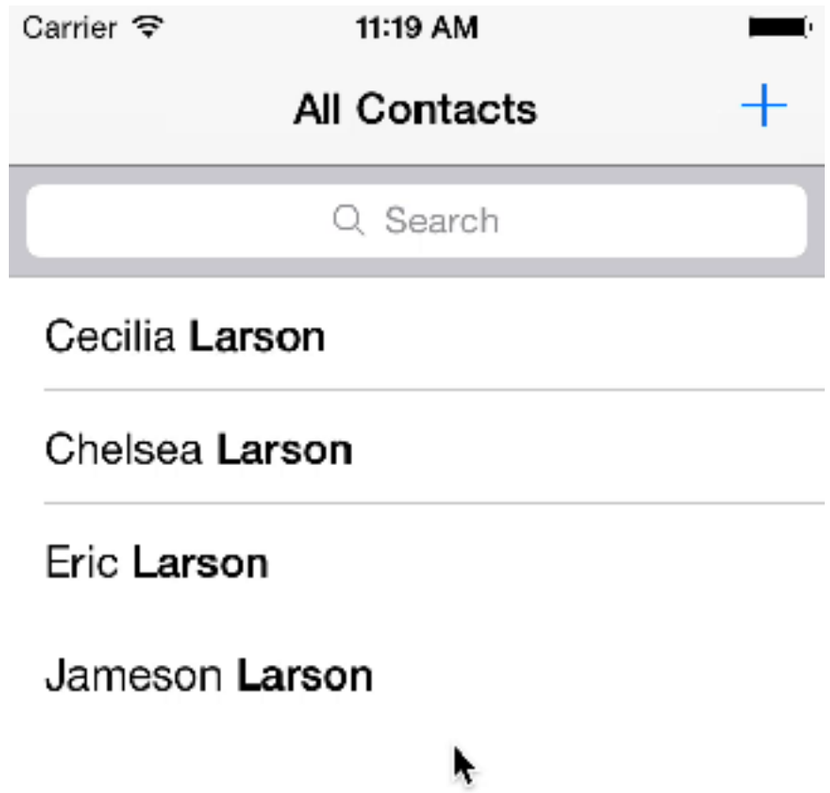
defining your style

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

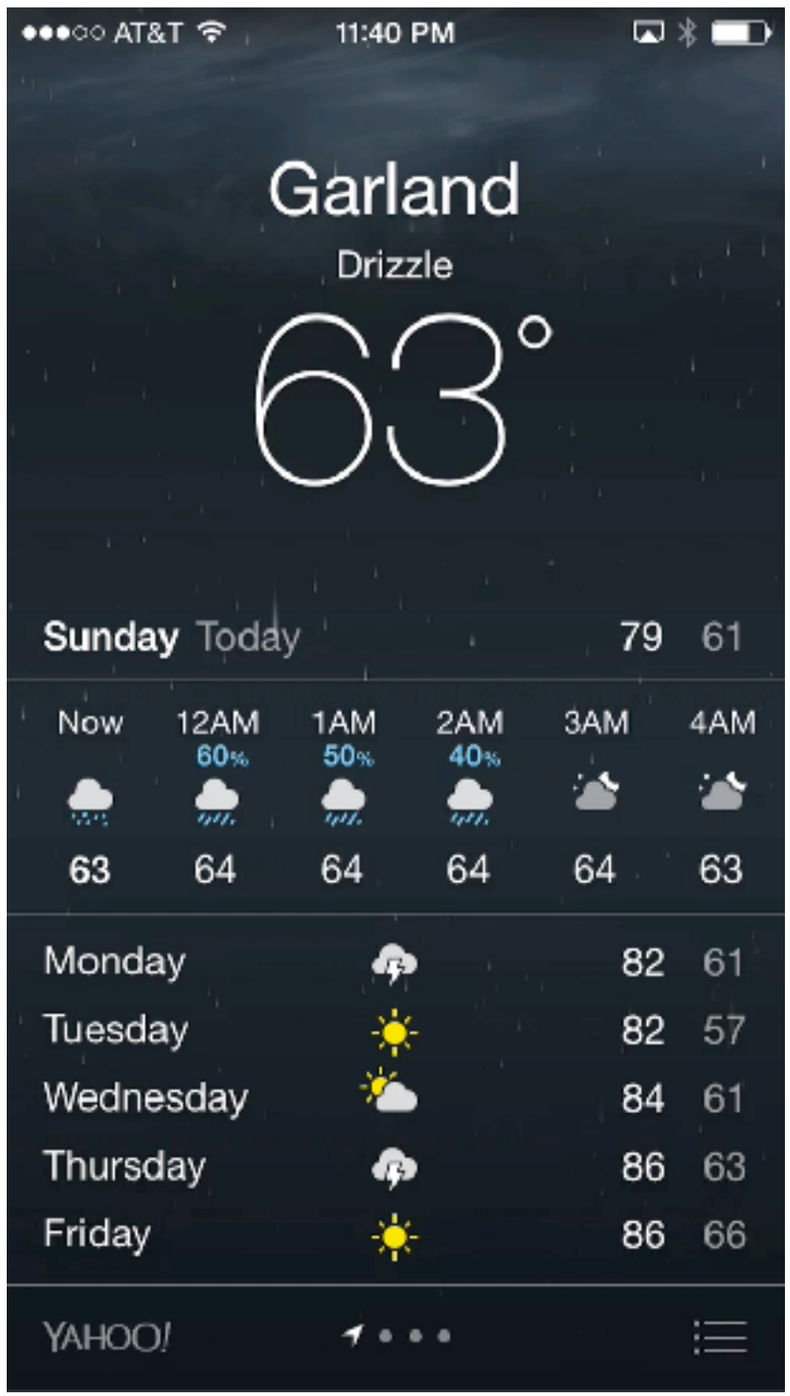
know your app: utility

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

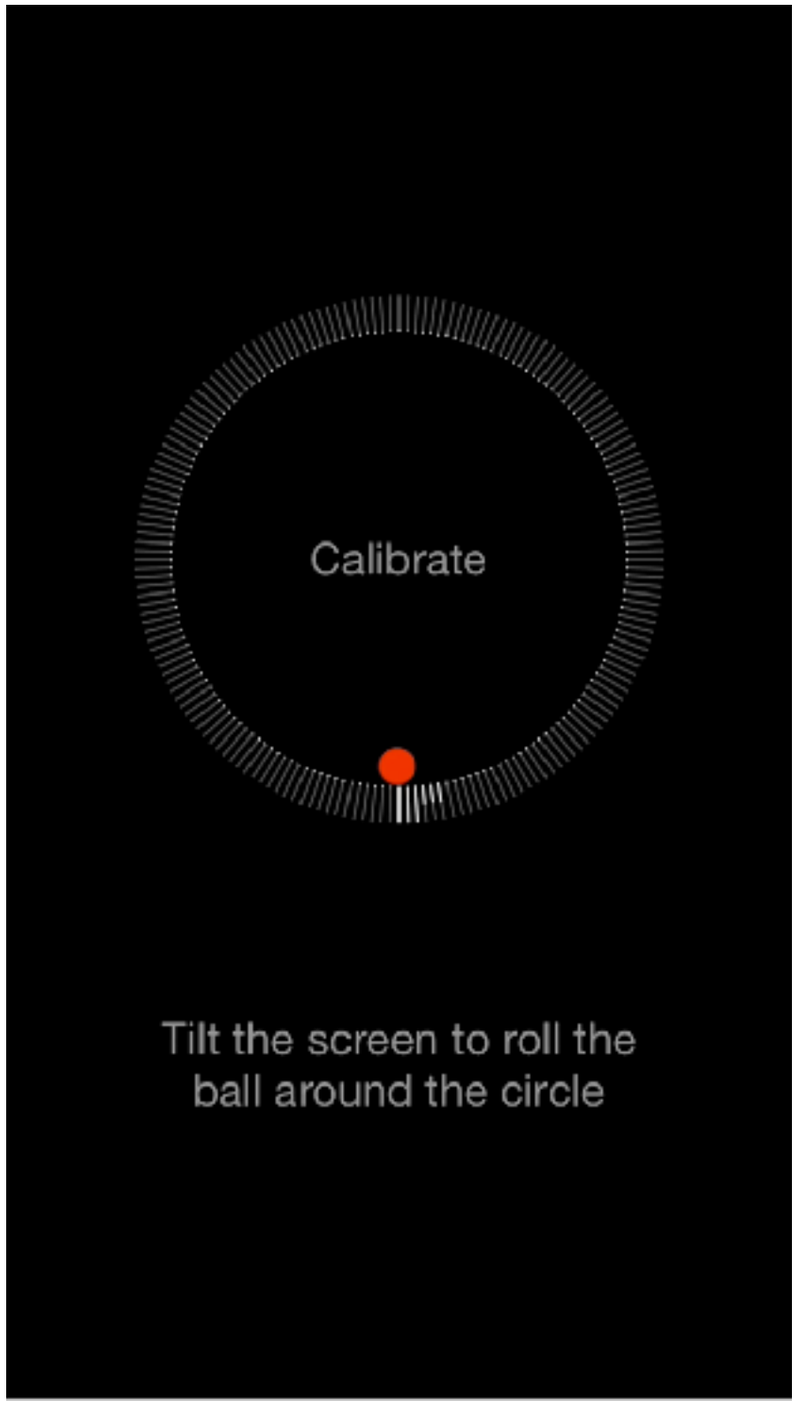


know your app: immersion

immersive

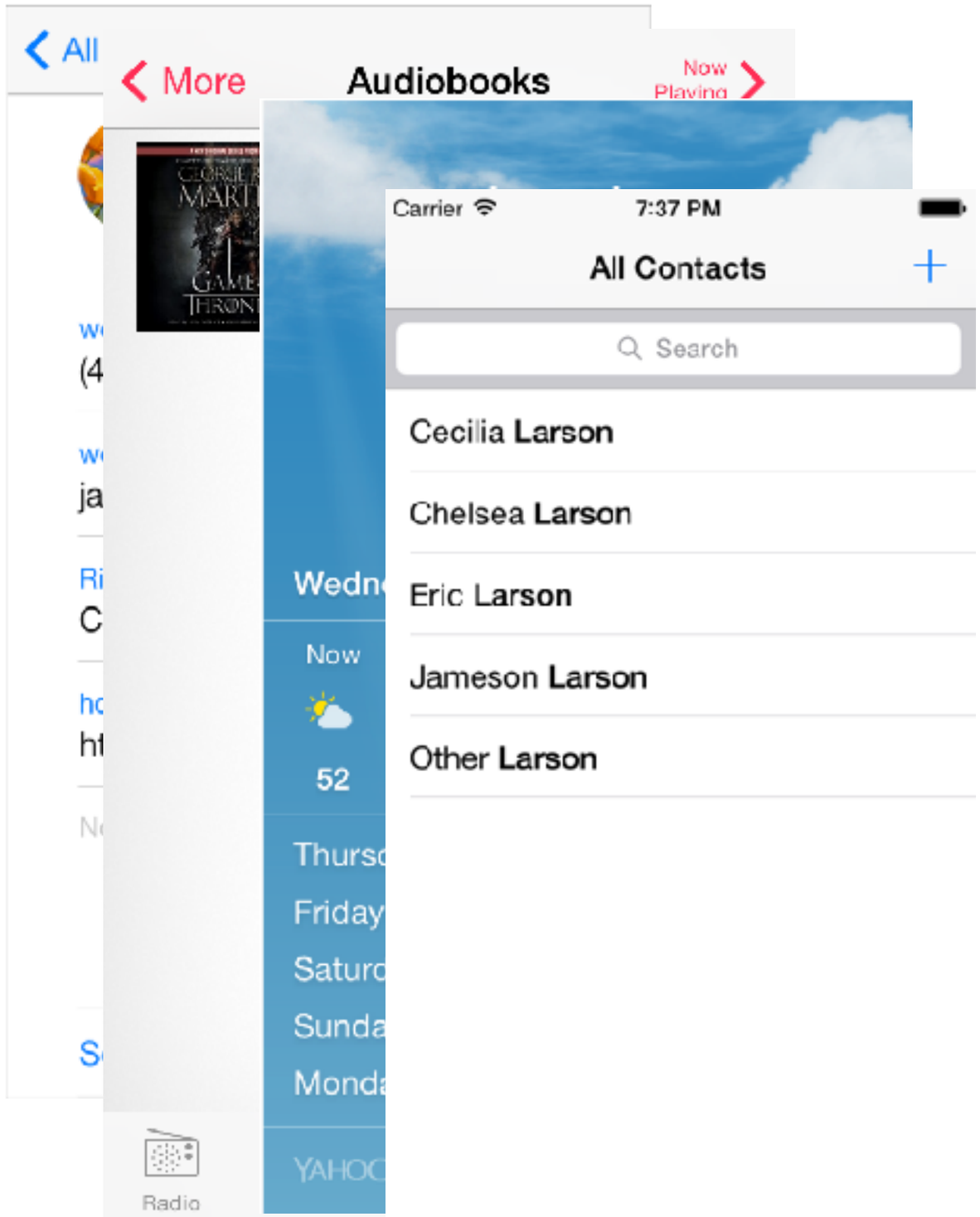
high interaction, visual experience

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



navigation is orientation

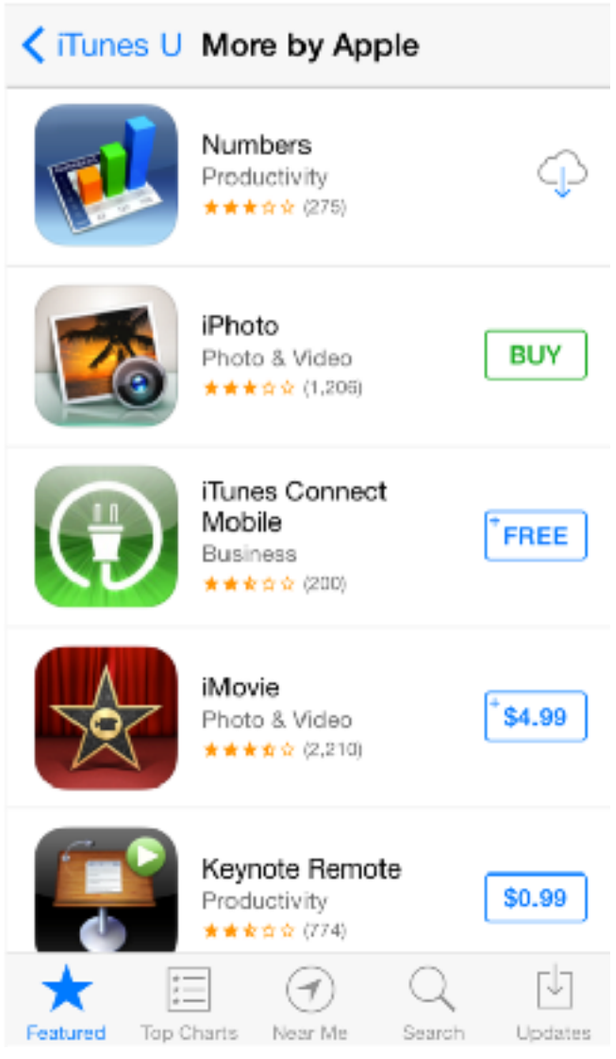
- the user must always know **where they are** in the app
- the **navigation bar** is the most understandable form of hierarchy, even when using animated hierarchy
- **tab bar** is for parallel content (peers)
  - each peer is different in function
- **page control** is for identical views, with new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



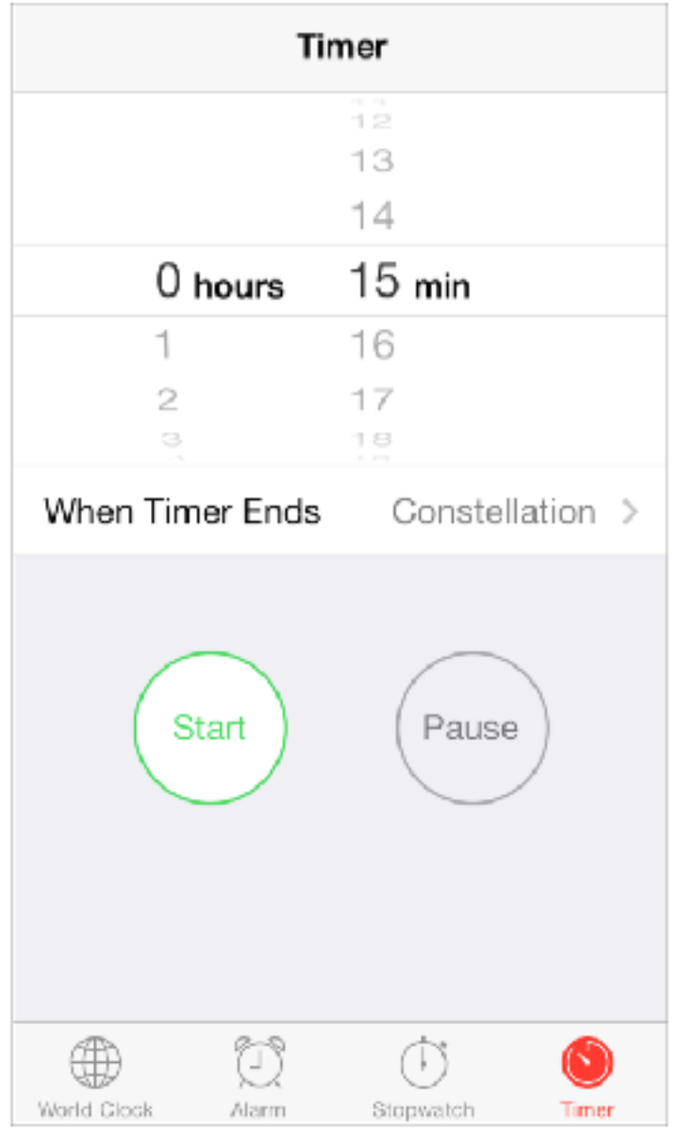
button design

- embrace the borderless
- until you need a border

distinguish “tap button” from “tap row”



most important interaction



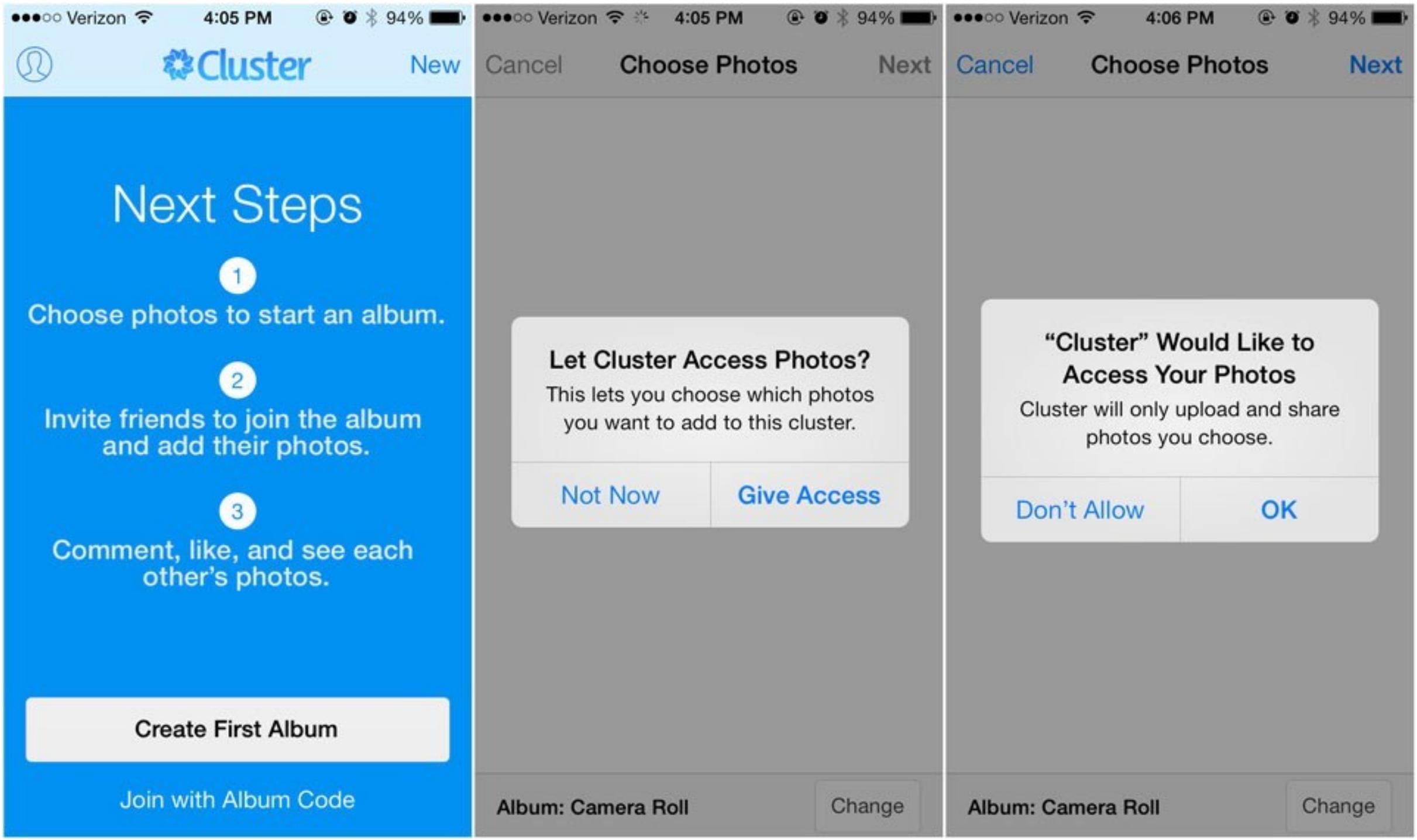
principles summary

- 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

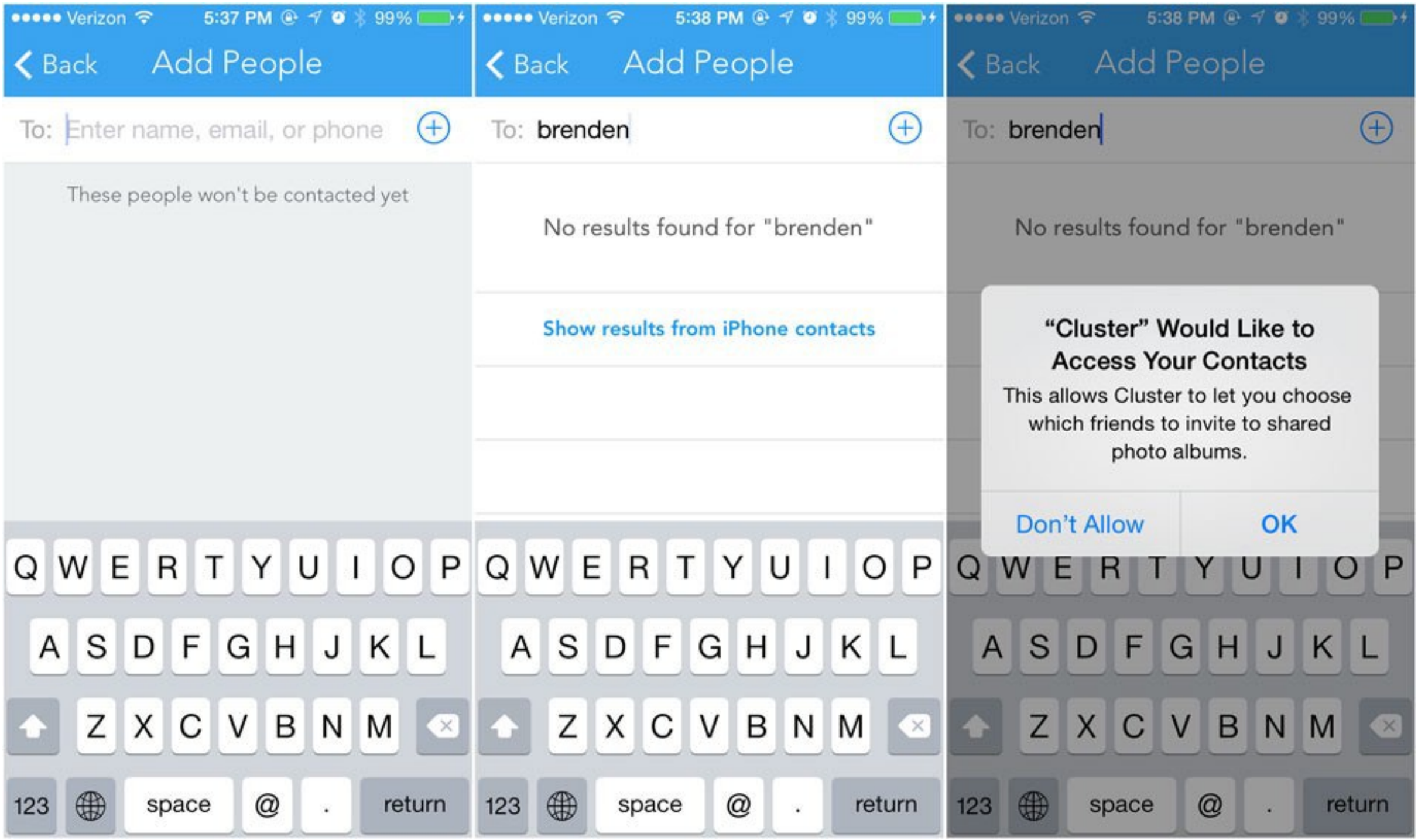
user control

- 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

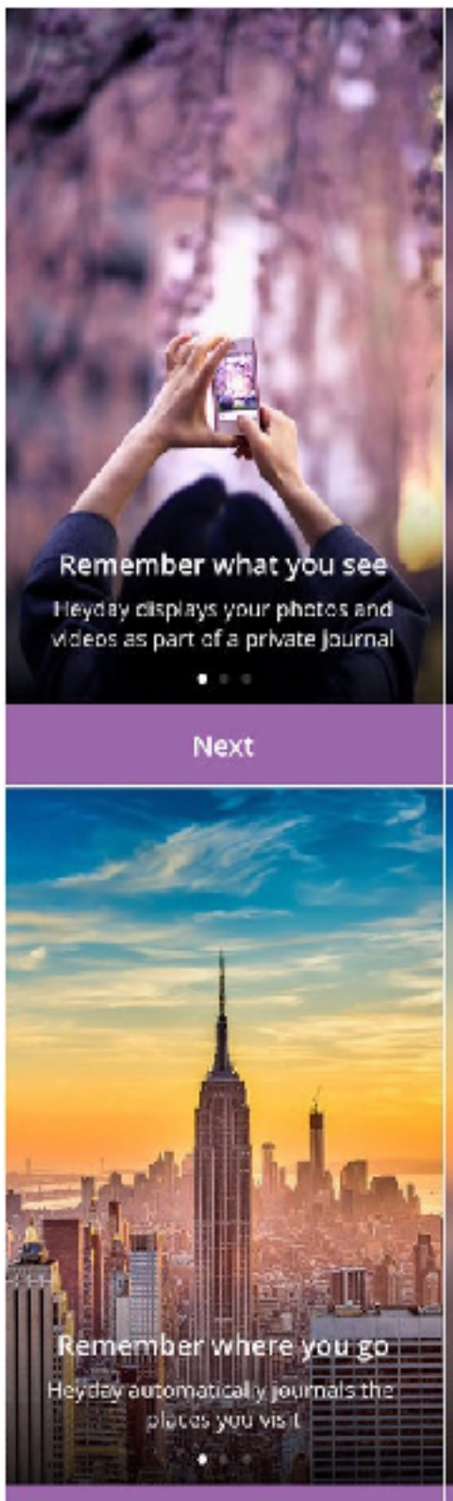
when asking for access

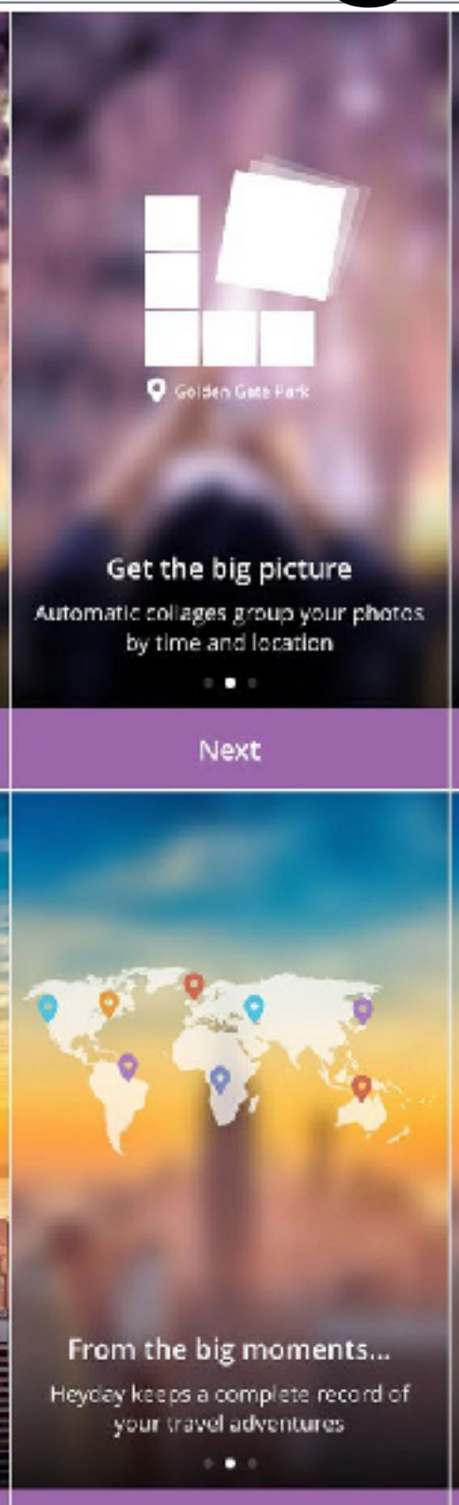


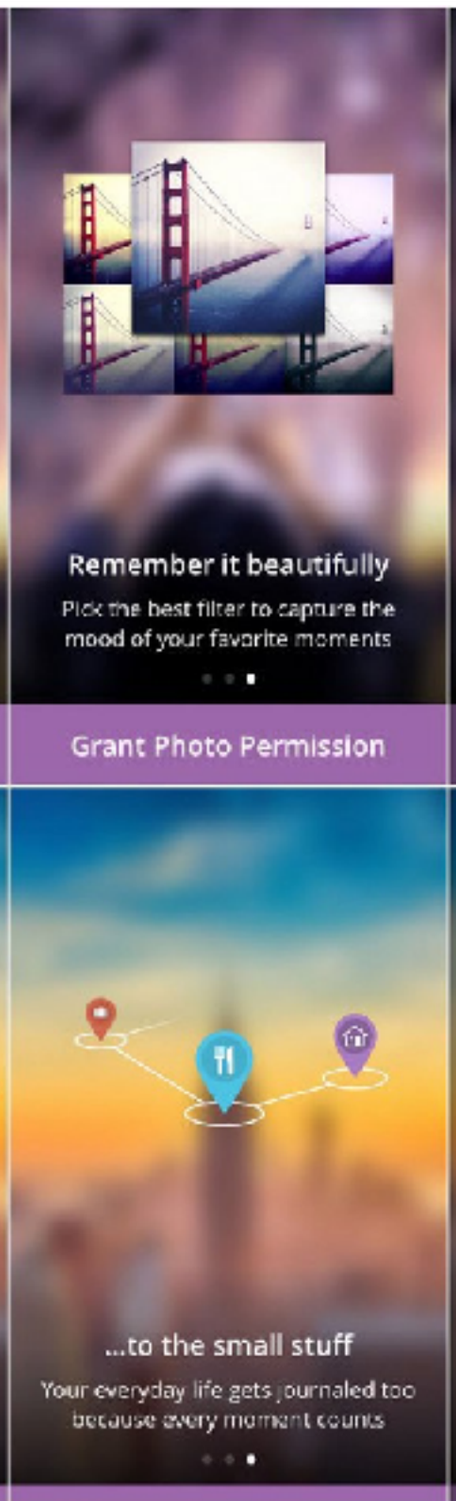
when asking for access

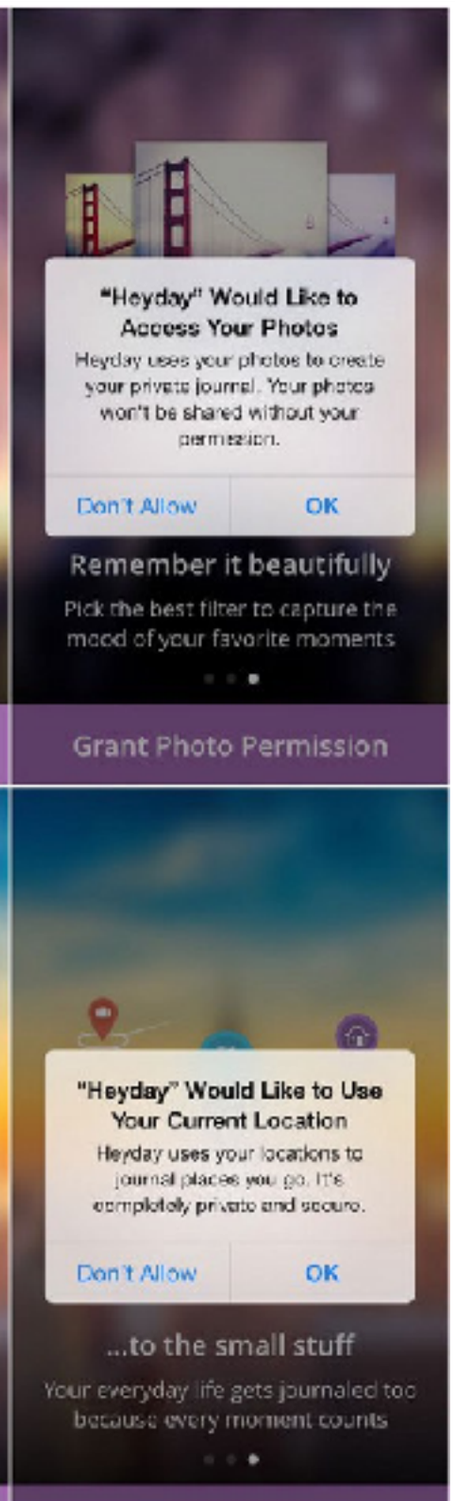


is this right?









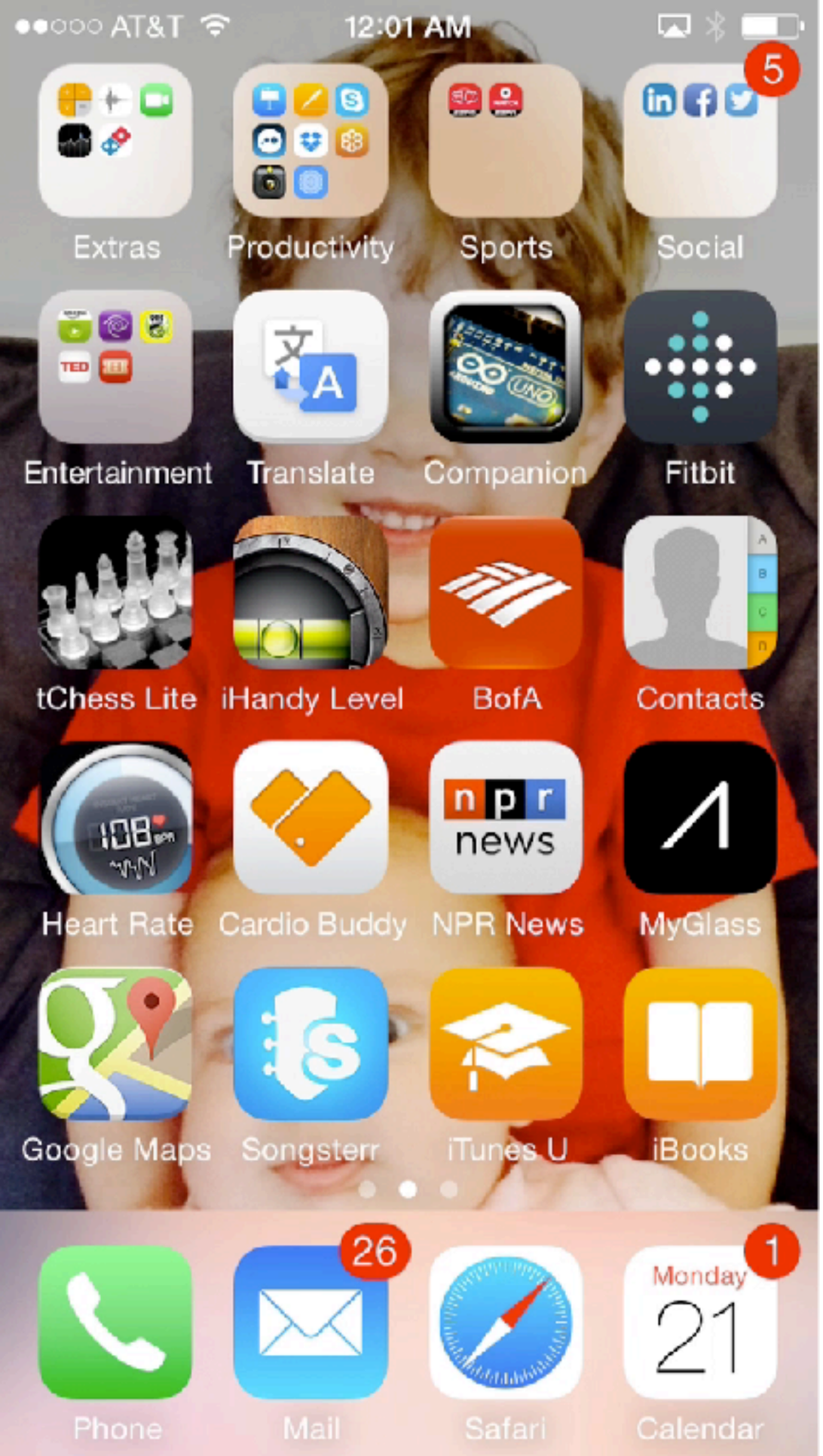
what is good?

what is bad?

some great examples

some great examples

an example

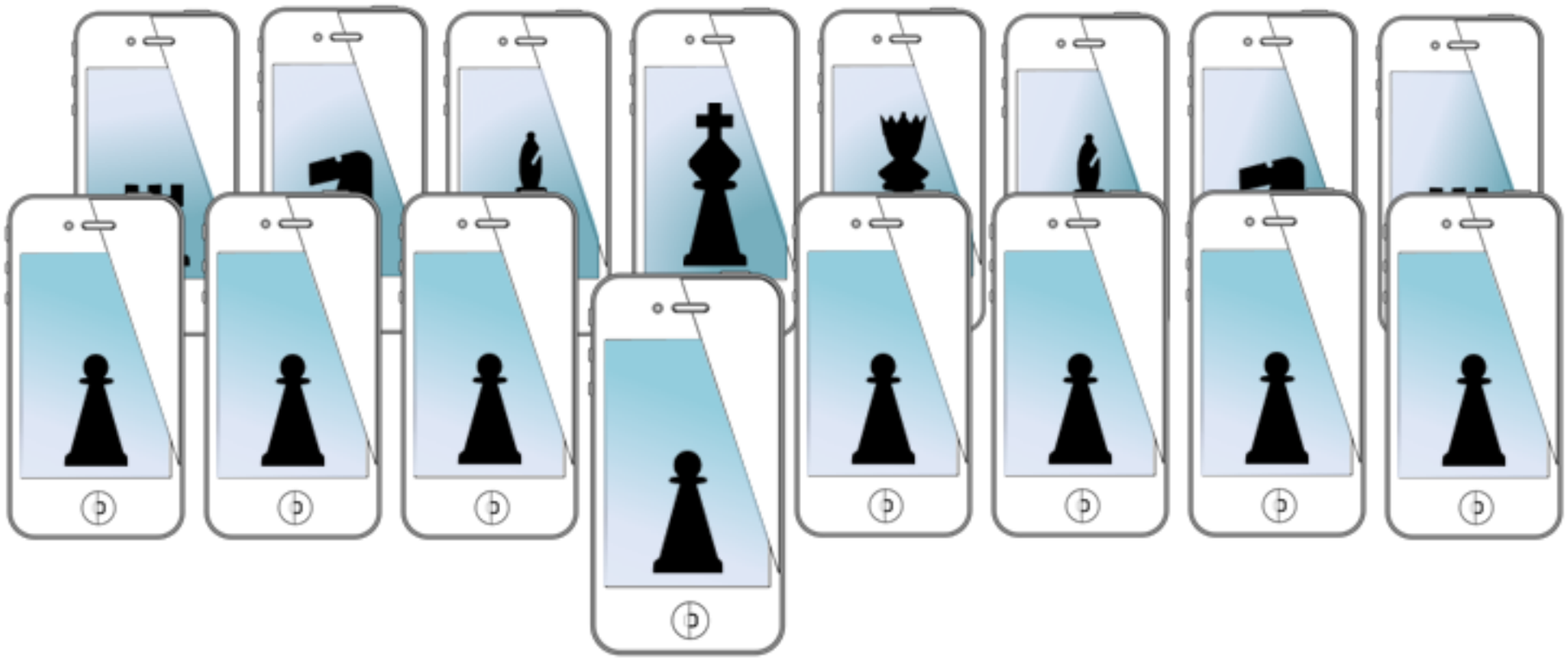


a **really** excellent example

for next time…

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

MOBILE SENSING LEARNING



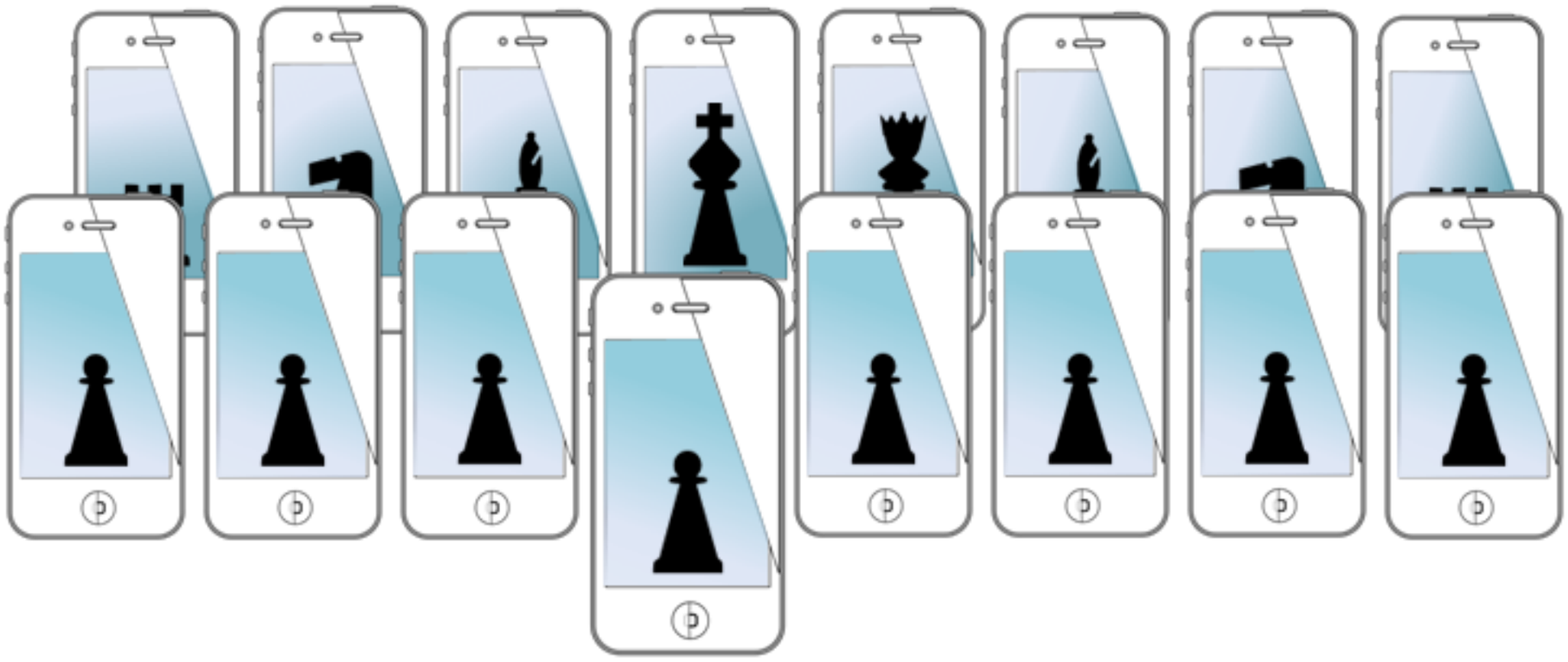
CS5323 & 7323

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

block name

param types

```
//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));
```

define code that will execute

Using a block within a method call:

enumerate with block

```
//------------------------------
//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]);
}];
```

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;
```

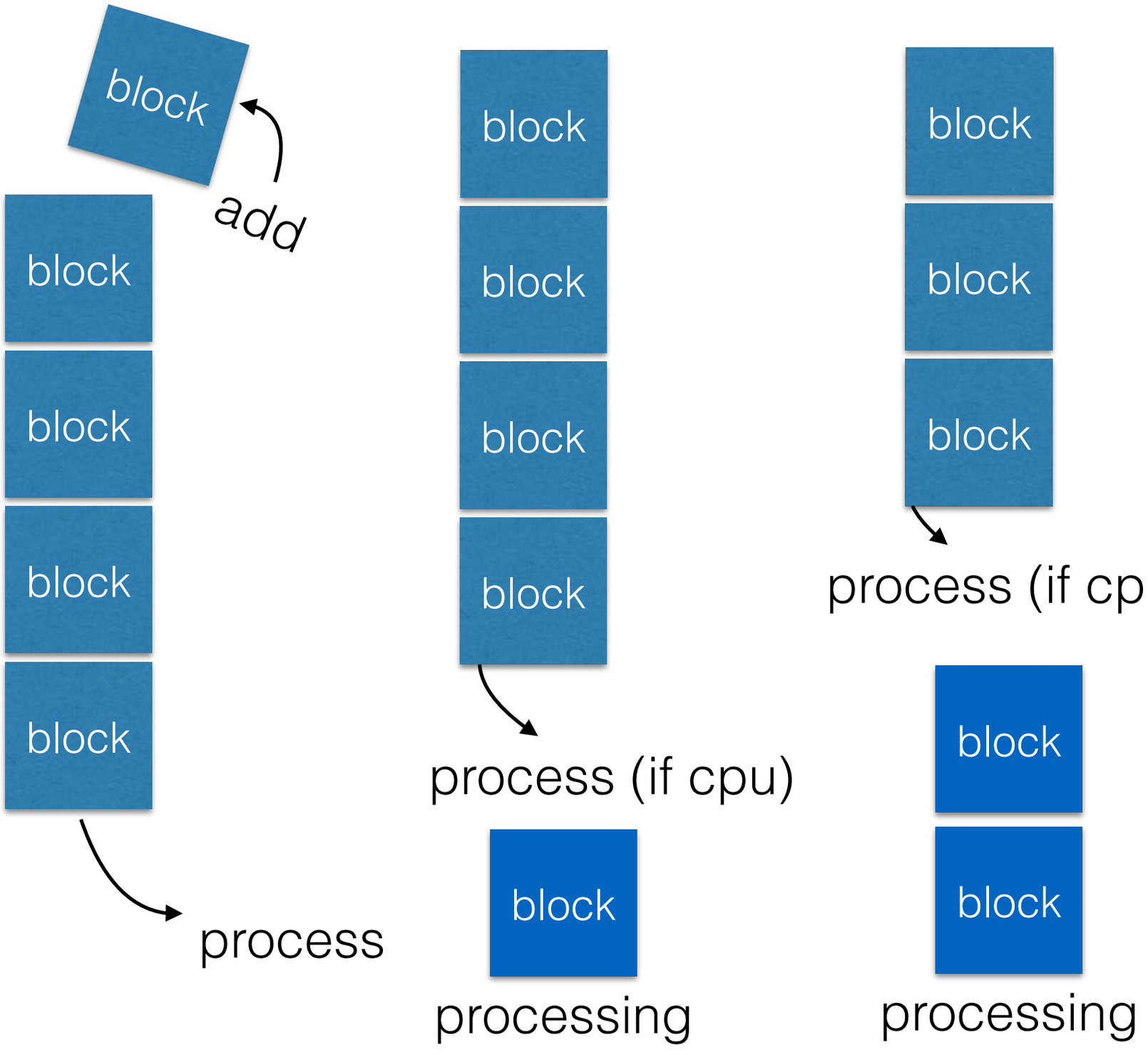
- classes hold a **strong** pointer to blocks they use
- blocks hold a **strong** pointer to __block variables
  - so using “self” would create a retain cycle

```
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**

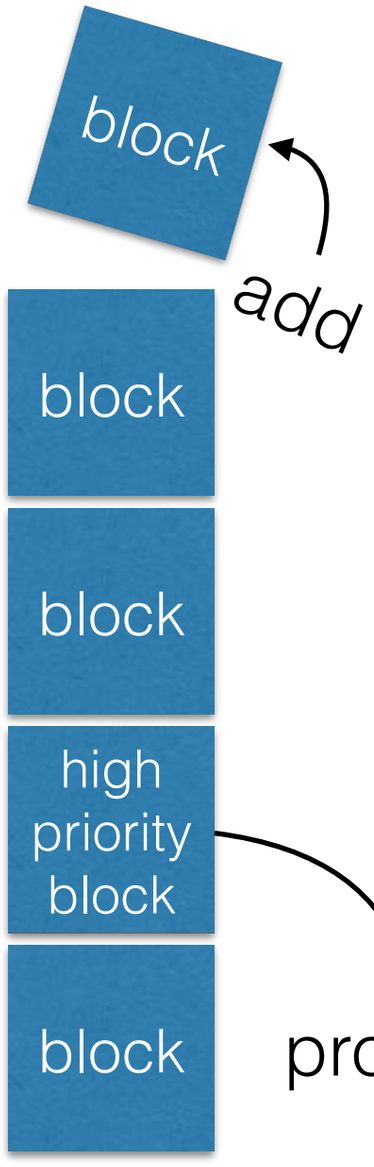
concurrent queues

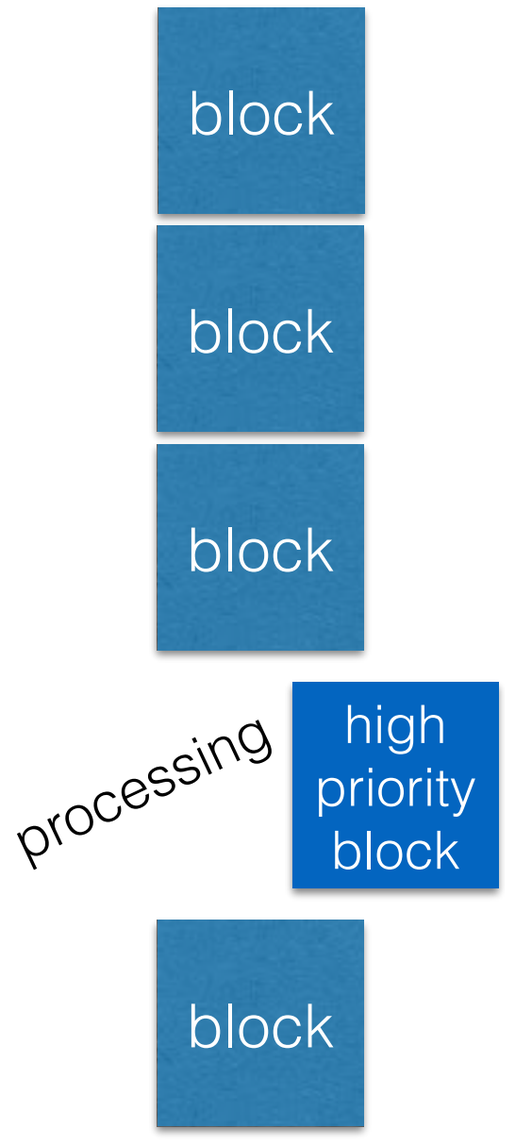


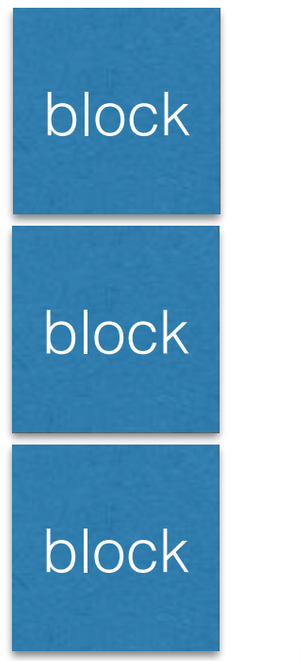
threads?

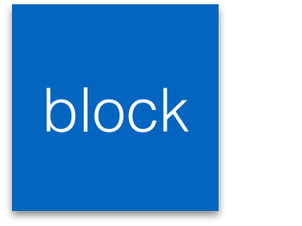
either could finish

the main queue









processing

one thread
UI is in sync

queue syntax

Legacy

```
// 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
```

create new queue

define block

serial or concurrent

update UI, main thread

Recommended

```
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];
    }];
```

create new queue

define block

update UI, main thread

queue syntax

- using global queues

access a global queue

```
// 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 not in the main thread!

    // call from main thread
    dispatch_async(dispatch_get_main_queue(), ^{
        self.label.text = @"Finished running from GCD global";
    });
```

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)

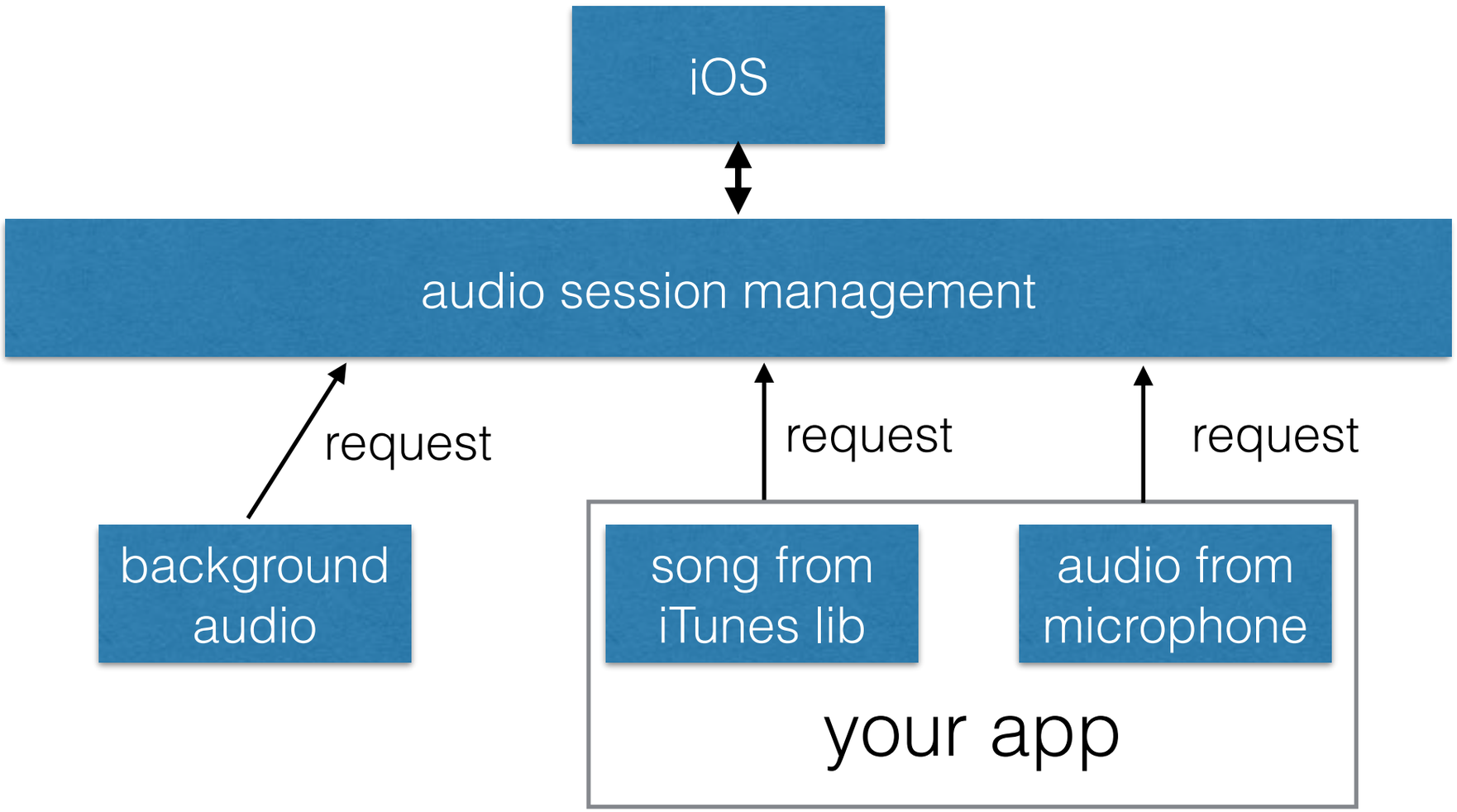
- this is no longer required for audio, but is good to know for later in the class when we use OpenCV…
- actually, its just c++
- …but need to tell compiler we are using c++

- 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

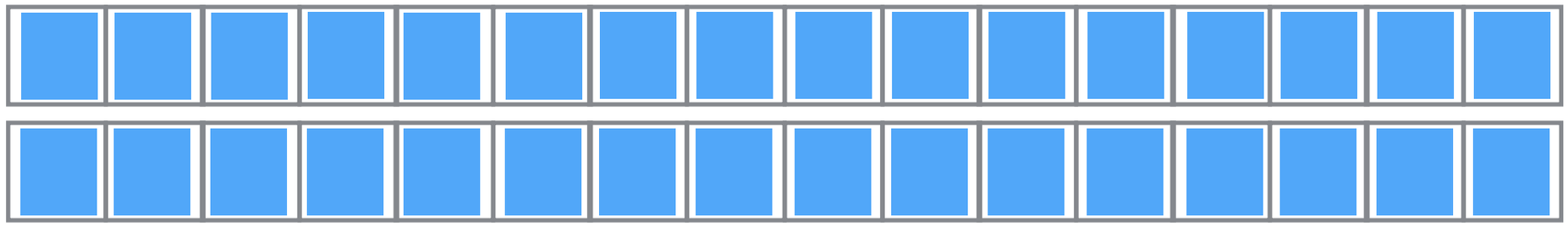


any request can alter management of other audio requests

mixing can be automatic — this is impressive

audio units

audio input buffer procedure, double buffer shown



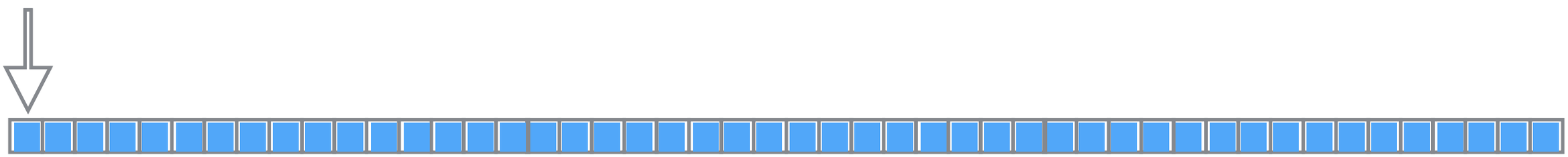
c

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!!



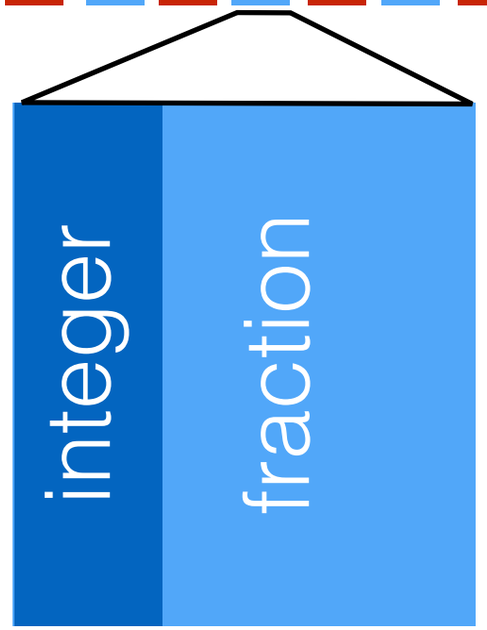
audio unit formats

microphone (input)

stereo could be interleaved (output)

right speaker

left speaker

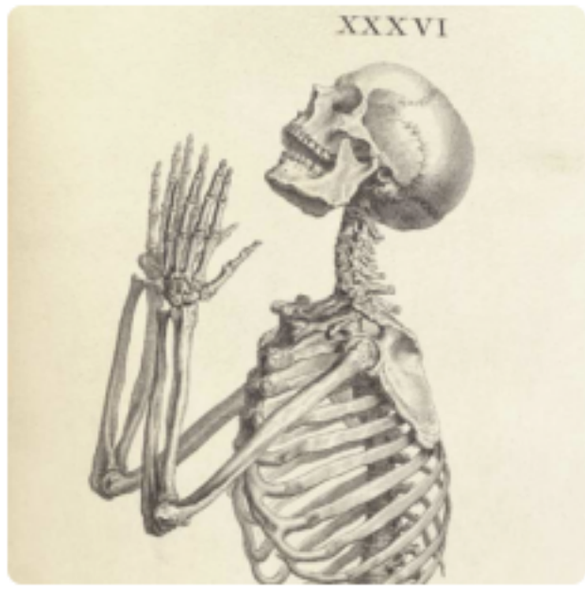


32 bits

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**

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

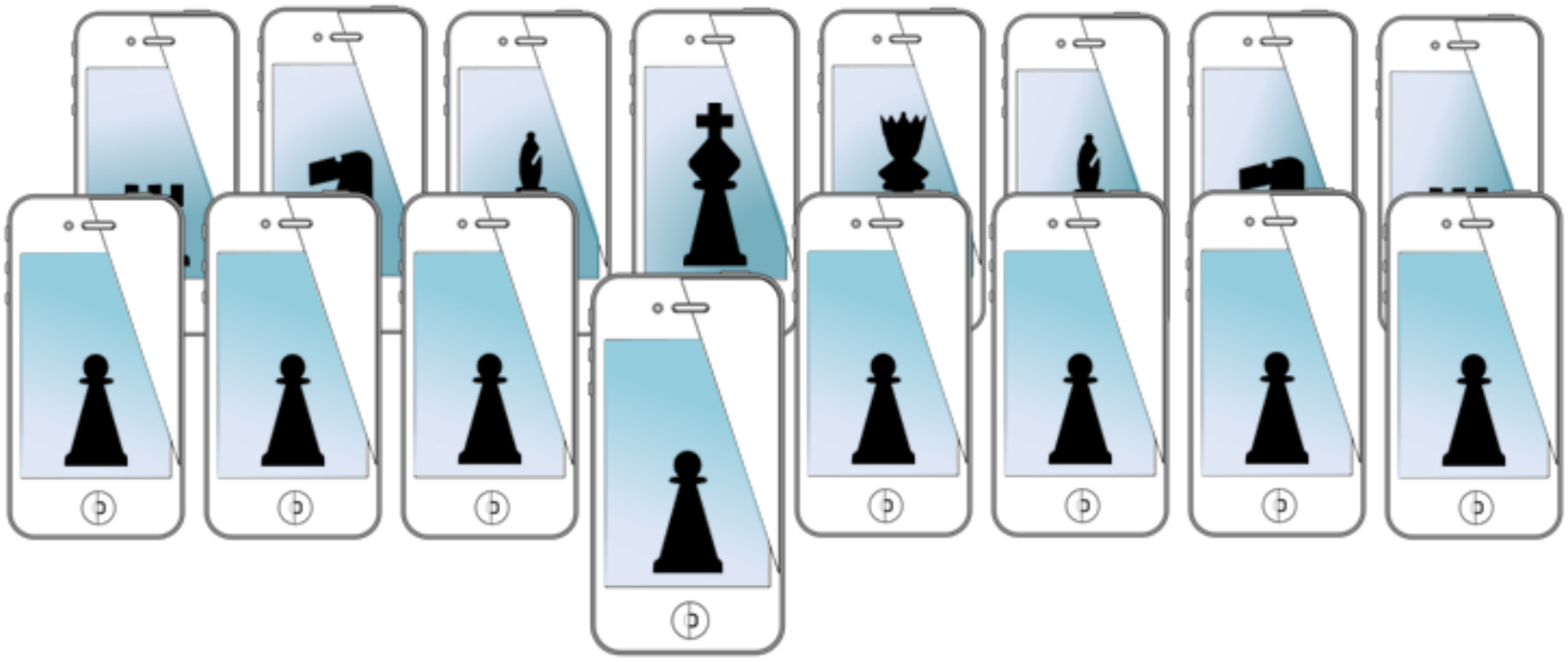
source code on GitHub

and 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