

# Microinteractions

## **Embedded Interface Design**

with **Bruce Montgomery**



# Learning Objectives

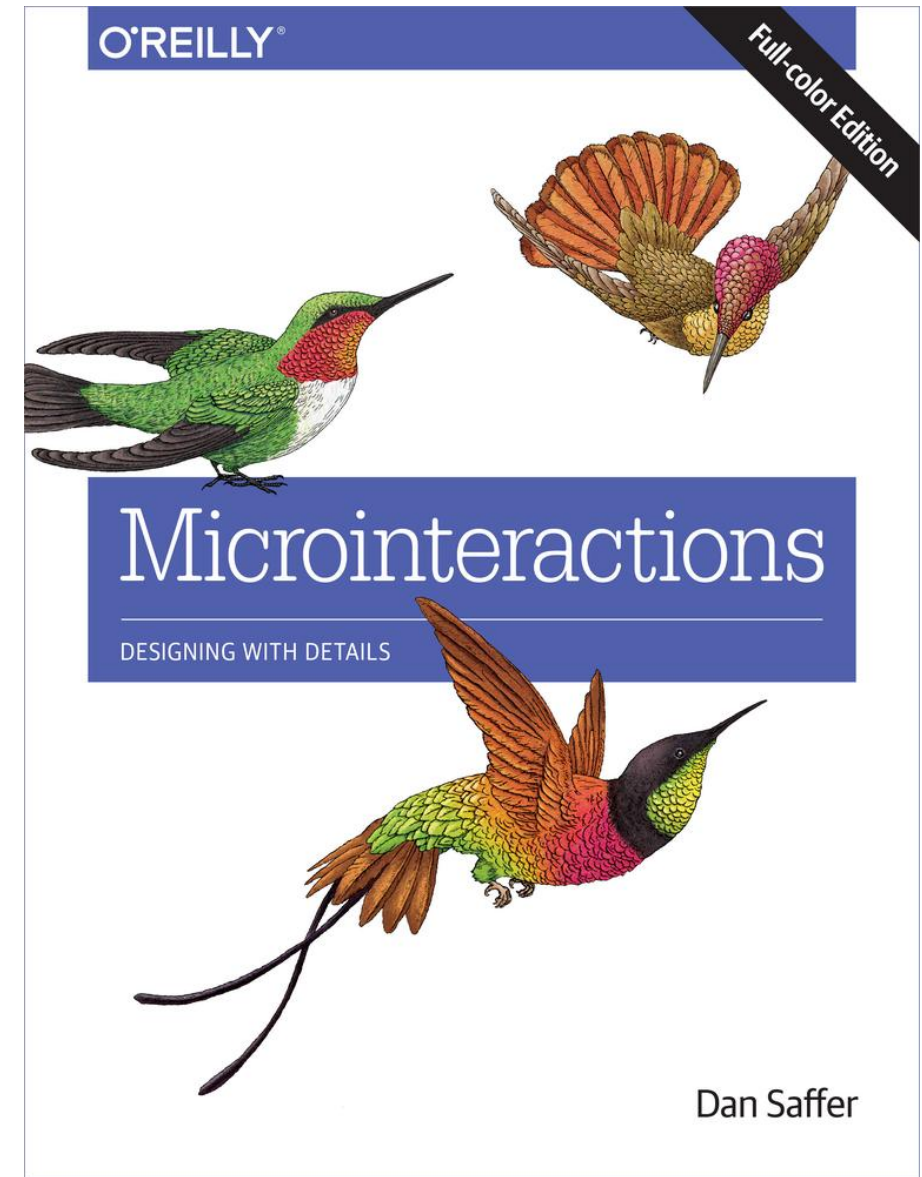
Students will be able to...

- Recognize the elements and definitions of microinteractions
- Consider applying microinteraction design to interface elements



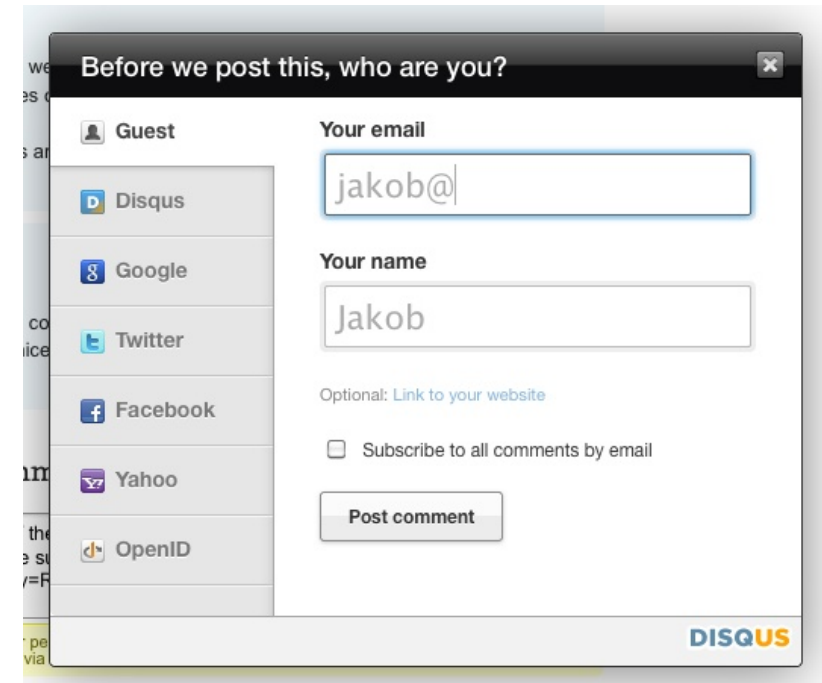
# Microinteractions?

- From a book by Dan Saffer
- O'Reilly, 2013
- A lot of buzz in the UX community when it came out
- Unless otherwise noted, material in this lecture is from this source



# What is a Microinteraction?

- A contained product moment that revolves around a single use case
- A tiny piece of functionality that only does one thing
- An introductory example is around the ring/silent switch on an iPhone and a new user – at a symphony performance
  - Should the ring/silent switch silence an alarm?
- Microinteractions are the functional, interactive details of a product
  - Details aren't just the details; they are the design -- Charles Eames
- Details make engaging with the product easier and more pleasurable – even if you don't consciously remember them



Another example – guessing your first name from your e-mail in a signup form...

# Little Big Details

- A website for capturing fine design details – with daily updates
- <https://littlebigdetails.com/about>
- Another example, Twitter's password entry process...

|          |                          |                                     |
|----------|--------------------------|-------------------------------------|
| Password | <input type="password"/> | ⇒ 6 characters or more (be tricky!) |
| Password | <input type="password"/> | ⇒ Too short                         |
| Password | <input type="password"/> | ⇒ Too obvious                       |
| Password | <input type="password"/> | Weak                                |
| Password | <input type="password"/> | Good                                |
| Password | <input type="password"/> | Strong                              |
| Password | <input type="password"/> | Very Strong                         |

# Microinteractions vs. Features

- Microinteractions differ from features in both their size and scope
- Microinteractions are simple, brief, and should be nearly effortless
- Features tend to be complex (multiple use cases), time consuming, and cognitively engaging
- A music player is a feature
- Adjusting the volume is a microinteraction inside that feature
- Microinteractions are good for
  - Accomplishing a single task
  - Connecting devices together
  - Interacting with a single piece of data, such as a stock price or the temperature
  - Controlling an ongoing process, such as changing the TV channel
  - Adjusting a setting
  - Viewing or creating a small piece of content, like a status message
  - Turning a feature or function on or off



# Microinteractions can be big, can be defining

- Devices with single interactions
  - A toaster
  - An app for unit conversion →
- Consider working with an OS
  - The individual applications the OS controls are large and multifaceted
  - The utilities and tools provided by the OS are more like microinteractions that define frequent simple transactions
  - Example: initial microinteractions from Android (compared to iOS) were poor
- “The design of your product is only as good as its smallest part”



# Importance of Microinteractions

- In competitive markets, microinteractions are how systems with feature parity differentiate themselves
- In multi-platform environments of various embedded devices, small interactions with small screens or physical controls focus attention on microinteractions
- Microinteractions include interacting through a fast glance, a rapid response, a quick review
- Designs are driven to be lightweight, low complexity, streamlined

and we'll get back to you real  
sure you get our response.

## Office hours

Monday through Friday  
9am-7pm Eastern Time

Right now it is 5:40pm at the office.

Example – the microcopy (tiny bits of text) that includes the time at the location as well as office hours.

Another example – the introduction of scrolling in SmallTalk in 1973-1976 from Alan Kay(!), Adele Goldberg, and Dan Ingalls at Xerox PARC





# Philosophy of Microinteractions

- Providing **Signature Moments** – interactions that are differentiating and unique
  - Ex: iPod scroll wheel, a signature sound, etc.
  - Builds customer loyalty, recognition, and brand strength
  - But still remembering this is not a feature design – “less is more”
- Reducing complex applications of a product to a product based on a key microinteraction
  - Additional microinteractions may create other products
  - A product that does one thing and does it well – a minimum viable product
- Looking at an overall design as large sets of microinteractions
  - Moving to the detail level but still maintaining an overall product view (big picture)
  - Making sure the overall system is coherent and cohesive
  - Always challenging for a system design driven by deadlines



# Structure of a Microinteraction



- Trigger – initiates the microinteraction
- Rules – how the microinteraction works
- Feedback – illuminates the rules
- Loops and Modes – meta rules that affect the microinteraction

# Triggers

- Triggers are the control that starts an interaction
- Can be physical or virtual (screen-based)
  - Example – phone silencing
- Can be user-initiated or system-initiated
- Example is a re-designed MetroCard subway vending machine based on a touchscreen →
- The trigger that starts an interaction shows a start button, but actually, the whole screen is the start button



# Trigger Design Principles

- What does the user want or need to do, when do they want to do it, and what is the context around their doing it?
  - UX research methods (observation, interviews, etc.) can help answer this
  - UX design and verification methods can help with the trigger design
- The trigger should initiate the same action every time
- Determine how visible the trigger needs to be
  - Can it be in a drop down menu or should it always be visible?
- Bring data forward
  - Ex: a count or progress element built into an icon
  - Ex: a stock value on a stock icon, etc.
- Clear visual affordance (should clearly indicate how it works)
- The more frequently used, the more visible – ease of discovery
  - Can be stimulus-based – the trigger gets the user's attention
  - Or goal-based – the trigger is searched out by the user
- Add a label only if it provides information the trigger cannot



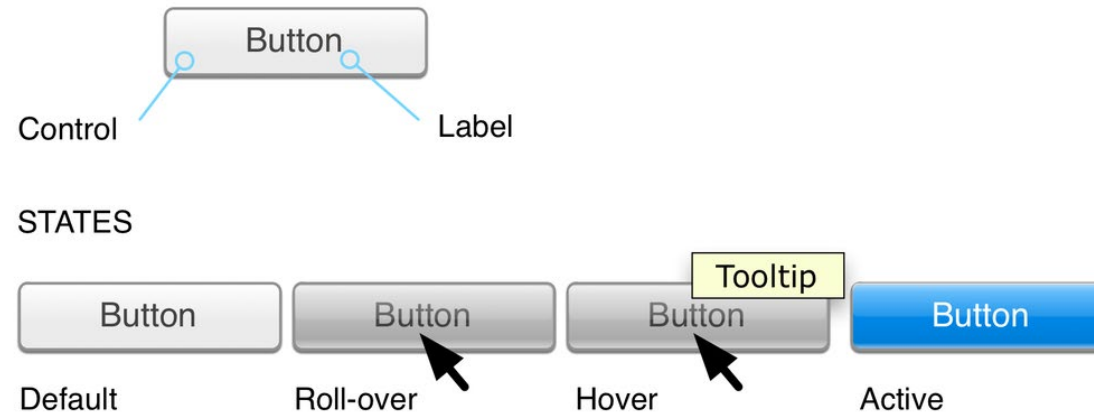
# The most discoverable triggers

- An object that is moving, like a pulsing icon
- An object with an affordance and a label, such as a labeled button
- An object with a label, such as a labeled icon
- An object alone, such as an icon
- A label only, such as a menu item
- Nothing: an invisible trigger
  - Sensors: touchscreens, camera, microphones, accelerometer, mouse move
  - Voice input
    - Always listening
    - Dialogue-based
    - Combined with a control
  - Try not to use invisible triggers for high priority microinteractions



# Components of a Trigger – Controls

- Controls
  - Single action – buttons or simple gestures
  - Two state (on/off) – toggle, checkbox, button with clear state change
  - Several states – dial (can include push/pull), sets of buttons, radio buttons
  - Continuous state – slider, dial, up/down buttons
  - Possibly multiple controls
  - Possibly custom controls
- Trigger components: the control itself, state of the control, the text or icon
- Parts of a control →



# System Initiated Triggers

- Causes
  - Errors
  - Location
  - Incoming data
  - Internal data
  - Other microinteractions
  - Other people/users
- Other considerations
  - How frequently should this happen
  - What is known that can make this more effective
    - Ex: showing a destination based on time of day →
  - How to show state changes
  - What happens with system errors?



# Rules

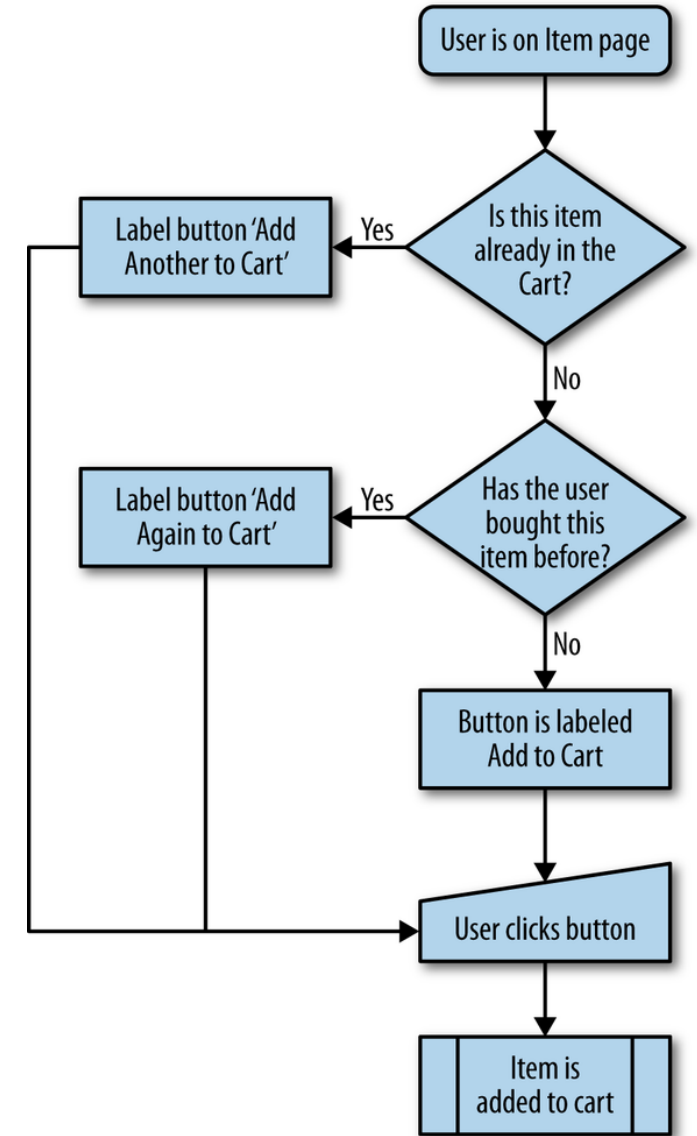
- Rules: How can the microinteraction be used? What is its model of operation?
- What is the goal? The goal is the engine of the rule
  - It is understandable and achievable?
- Rules determine:
  - How to respond to a trigger activation
  - What control does the user have to something in progress
  - What is the sequence of actions and the timing
  - What data is being used from where
  - Configuration and parameters of algorithms
  - Feedback delivered and when?
  - What mode is the microinteraction in and when
  - Does this repeat, how often?
  - What happens when this ends?
- Revealed to the user in what can be done and what can't





# Developing Rules

- Identify the main actions that have to be performed in order
- Details the elements as you design
- Good place for a sequence diagram →
- Actions as verbs; objects and states as nouns
- Best interactions allow a variety of actions with a limited number of objects
- Use progressive disclosure to reveal only what is necessary at that moment to make a decision or manipulate a control
- Objects to interact with generally have three states
  - Invitation/default state
  - Activated state
  - Updated state



# Constraints for rules

- Rule constraints (technical, business, environment, etc.)
  - Available I/O devices – screen, keyboard, speakers, etc.
  - Type or range of input
  - Expense – in resources or services
  - Available data
  - Data that can be collected
- Examples of available data – used to predict or enhance microinteractions
  - Device being used
  - Time of day
  - Noise in area
  - Time since last microinteraction
  - Is user alone or with others
  - Battery life
  - Location/direction
  - Past actions
- Tesler's Law – Law of Conservation of Complexity
  - All activities have an inherent complexity – which is handled by the user or the system
- The microinteraction should use data to remove choices or actions that are not valid or unlikely
- Better handled by the system:
  - Computation/calculation
  - Multiple simultaneous tasks
  - Remembering things
  - Detecting patterns
  - Searching through data



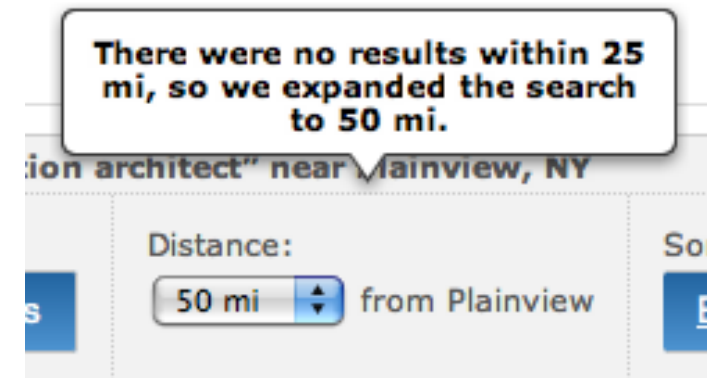
# Rule Design – Limiting Options

- Limiting options and using smart defaults
  - Emphasize the next action a user is likely to take
    - Helps link microinteractions
  - Be ruthless in eliminating options
    - Ex: Google's search box – most effective microinteraction of the early 21<sup>st</sup> century
    - More than one major option is likely too many
  - Most prominent default should be most likely next action
  - Avoid non-meaningful choices
    - If it doesn't enhance value or pleasure, leave it out
  - Reduces edge cases for testing/behavior



# Controls and User Input; Preventing Errors

- Choosing between operational simplicity and perceived simplicity
- If it will be done repeatedly, perceived simplicity is important
  - Availability of shortcuts, for instance
- If the microinteraction is a rare one, keep it operationally simple – require as little foreknowledge as possible
  - Examples: text fields that can adapt to input variations (phone numbers)
- Error displays should ideally be limited to system issues, not user input
- Microinteractions should try to fix any errors they may encounter →
- Should also prevent users from using a control in an unintended fashion



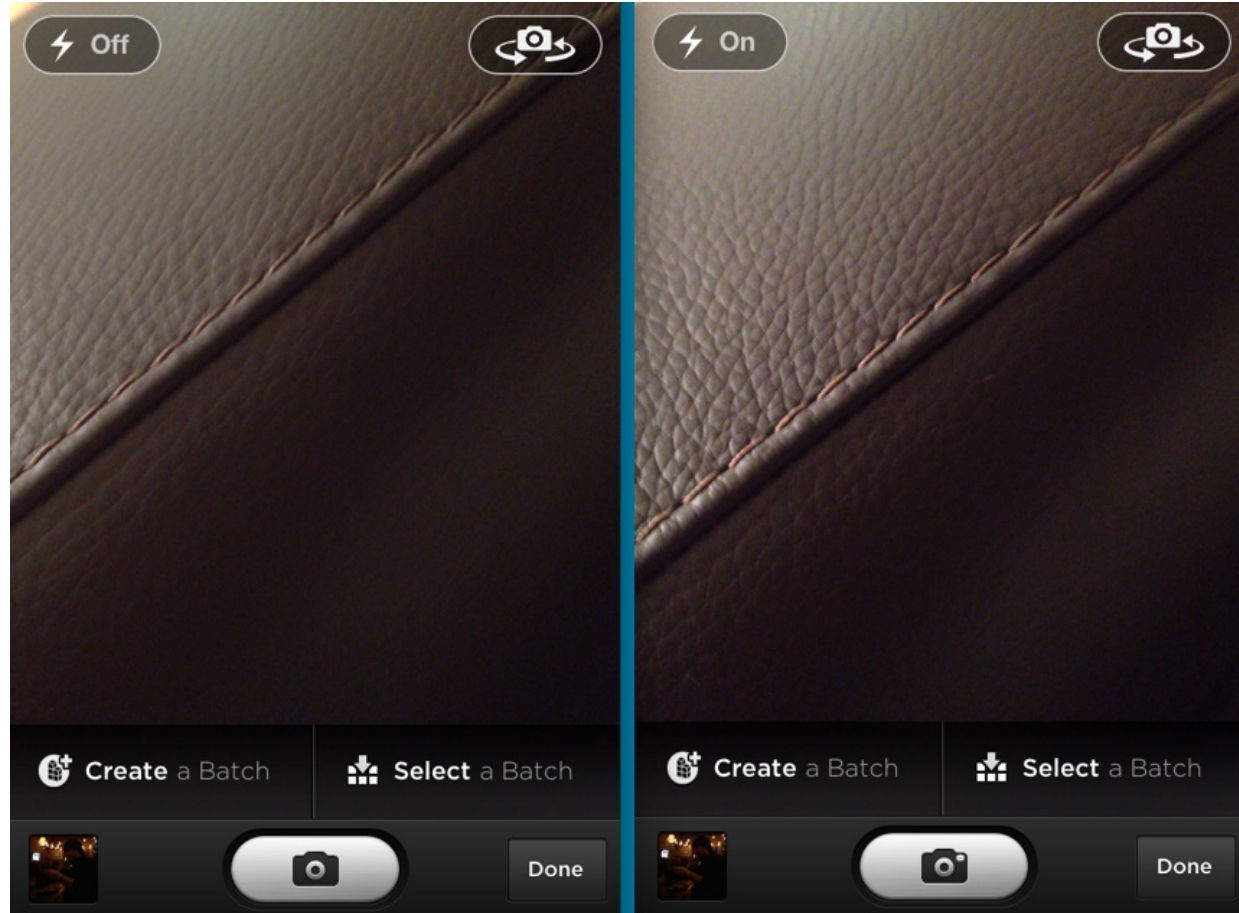
# Microcopy Rules

- Make sure the text is needed, keep it as short as possible
- Never use instructional copy when a label is enough
  - If a word doesn't fit – consider an icon and text tooltip
- Avoid easily misinterpreted text
- Location of text
  - Best is above, second best is on or in the object (interesting as traditionally, icon text is placed below)
- Labels in text fields disappear, and users may forget what the field is for
- If you use instructional text, ensure all text matches
- For time/date, users prefer relational dates/times (“three hours ago”) to an exact time that makes the user do the calculation



# Feedback – Illuminating Rules

- Do not overburden users with feedback – what is the least amount of feedback to convey what's going on
  - Ex: Flash on/off – the camera button
- Driven by need
- Visual, audible, haptic, other
- Feedback rules:
  - Change based on context
  - Duration
  - Intensity
  - Repetition



# When should Feedback occur?

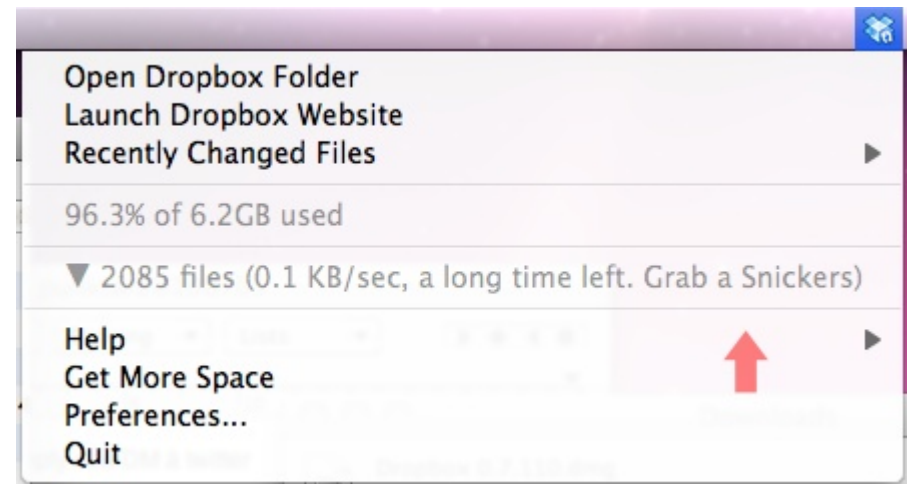
- When do we need feedback?
  - Immediately after a manual trigger or a manual change to a rule
  - On an system-initiated trigger when state changes significantly
  - Whenever a user reaches (or exceeds) the edge of a rule
  - Whenever the system fails to execute a command
  - Showing progress on any critical process, especially long running ones
  - Could occur at the beginning or end of a process, or when changing modes
- Remember, feedback is for people
  - Something happened
  - You did something
  - A process started or ended
  - A process is running
  - You can't do that
- Feedback can't always be text
  - Sound, icons, color, images, light, haptics





# The best Feedback...

- Never arbitrary – actions should be clearly connected to responses and results
- Less is more – convey the most information with the least amount of feedback
- The more important feedback is, the more prominent it should be (and possibly shown in multiple ways)
- Use overlooked parts of the display as a means of extending message delivery
  - Scrollbars, cursors, progress bars, tooltips
- Feedback can be a personality opportunity for a microinteraction (add edge or humor) →
  - Avoiding the “uncanny valley” of discomfort





# Feedback - Visual

- If at all possible, user-initiated action should be accompanied by visual feedback
  - System-initiated actions only need this for requiring human intervention
- Try not to make visual feedback redundant or intrusive
- The feedback should occur as close to the point of user input as possible
- Text should be direct, short, clear, and human
  - Avoid personal pronouns or derisive terms – “you” or “user”
  - Avoid identifying device as a person – “I” or “me”
  - Focus on the specific action that needs to be taken



NOV



**Questar Gas Company**

**04**

*Check will be withdrawn tomorrow, mailed Monday, and arrive between the 7th and 9th.*

# Feedback - Animation

- Animation should be used sparingly, and only to communicate something to the user
- If used, animation should be:
  - Fast – not a delay to the activity
  - Smooth – choppy motion will make the activity seem broken
  - Natural – obey gravity, inertia
  - Simple – meaningful and understandable
  - Purposeful – not just for decoration
- Typical reasons for use
  - Maintaining context while changing views
  - Explaining what just happened
  - Showing relationships between objects
  - Focusing attention
  - Improving perceived performance
  - Providing a transition
  - Encouraging engagement



# Feedback - Audio

- Audio should be used sparingly, but is useful for devices without screens or hands-free
- Two primary uses – emphasis and alerts
- The Foghorn Test: Is this action important enough that users need to be aware of it when they cannot see it?
  - Even if it is important, you'll likely want to allow it to be disabled
- Two kinds of audio: words and “earcons” – short, distinct sounds meant to convey information
- Ideal earcons are those users relate to easily to make associations
- Avoid sounds that are too shrill or soft
- Duration is usually significantly less than a second (except for ongoing process)
- Signature sounds are often two to four notes in quick succession
  - But most microinteractions earcons should be a single pitch sound played once
- Speech input and output is more common, and almost required for dynamic messages
  - Be aware some people actively dislike it (your instructor, for one)



# Feedback - Haptics

- Haptics – or vibrotactile feedback
- Mostly felt, sometimes heard
- Sense of touch is somewhat limited
  - a claim that it's 1% of the information transfer of hearing
  - most people can detect only three or four vibration levels
- Uses
  - Enhance a physical action
  - Alert without sound
  - Artificial touch sensation (ex: with scrolling)



# Loops and Modes

- Loop – command or series of commands repeated for a set duration
- Mode – a fork in rules
  - Generally should occur sparingly in microinteractions, which are mostly modeless
- Common case – a settings mode – separate from the interaction itself
- Ideally, during microinteractions, modes should be avoided, as users may be confused by which mode they're in
- Microinteractions should generally have their own presentation or screen
- Spring-loaded modes – a mode that only occurs when a control is active – ex: shift key
- One-off modes – a mode that lasts for one action and then turns off – ex: iOS cut/paste
- Loops
  - Count-controlled, condition-controlled, collection-controlled, infinite
  - Open (no response to feedback), Closed (self-adjusting with feedback)
- Progressive disclosure or reduction based on repeated use – “the long loop”



# Example: Dishwasher Control Panel

- The user loads the dishes and detergent into the dishwasher, then shuts the dishwasher door
- Unless Reset has been used, the last cycle used and accompanied by estimated duration on the Progress Bar should be lit up and the Start button should pulse (Push Me state) until pressed
- The user can change the washing cycle, which changes the duration on the Progress Bar
- The user presses the Start button; the Start button glows red (Working)
- The dishwasher starts washing the dishes; the LED progress bar counts down
- If the dishwasher is opened, pause the cycle; when re-closed, resume; if the door remains open for more than an hour, reset
- When the dishwasher is done, the cycle button and progress indicator turn off; the Start button glows green
- When the dishwasher door is opened, the Start button switches to off
- At any time, if the user presses and holds the Start button for three seconds, the microinteraction resets and dishwashing stops; all buttons go to the off state and the Progress Bar is cleared



# Fixing a dull Microinteraction

- Should this be a Signature Moment?
- Am I starting from zero?
- What is the most important data in this microinteraction, can I bring it forward?
- Would a custom control be appropriate?
- Am I preventing human errors?
- Am I using what's overlooked in the UI?
- Can I make an invisible trigger for advanced users?
- Are text and icons human?
- Can an animation make it less static?
- Can sound, haptics, or other channels of feedback improve it?
- What happens when the user returns the second time or the 100<sup>th</sup> time?



# Observing a Microinteraction

- Too many clicks, taps, control presses? = too much effort
- Confusion as to why? = misnamed label, vague process, etc.
- What just happened? Did anything happen? = missing feedback
- Can't find what I'm looking for? Where am I? = labeling or mode issue
- What did you do to my data or input? = expectation not matching outcome
- If I do this, what happens? = poor label or instructions
- I didn't see that button or text. = visual hierarchy/positioning issue
- I didn't know I could do that. = action is too hidden
- What do I do now? = unclear next step/path
- What am I seeing there? = unclear feedback on process





Go to **www.menti.com** and use the code **87 69 81**

**Please enter your Identikey for participation credit (mine is brmo3998):**

 Mentimeter



Slide is not active

Activate

Pause scroll

 0

# Class content discussion

- Class/Instructor Intro/Syllabus
- Tools
  - Python, Node.JS, SMUD Usability Study
  - Git, Dev Env, Multithreading/Multiprocessing
  - QT, Data Stores – Project 1
- M2M & IoT Communications
  - M2M/IoT, Low Level Protocols, IoT Protocols
  - M2M Protocols, LPWAN Protocols
  - Cloud Architecture, Cloud for IoT, AWS, AWS IoT, Alternatives to AWS
  - HTML, IoT Security – Project 2
  - Message Queueing, APIs/Swagger, Microservices



# Class content discussion

- Projects
  - Project 1 – DHT22 w/MySQL to QT Client
  - Project 2 – Tornado and Node.JS Servers to HTML Client
  - Project 3 – AWS IoT, Lambda, SQS, SNS
  - SuperProject 4/5/6
    - Magic Wand – AWS Voice to Text, Text to Voice, Image Processing, SQS, API
    - Or your choice
- Exams
- Quizzes
- Dropped homework and added it to projects
- Extra Credit
- In class activities
- Other?



# STMicro workshop in Longmont on Wednesday vs. In-class Hackathon

- Free STMicro sensor workshop from 8 AM to 3:30 PM in Longmont (you have to register)
- [https://www.st.com/content/st\\_com/en/about/events/events.html/sensor-ecosystem-workshops-ame-2019.html](https://www.st.com/content/st_com/en/about/events/events.html/sensor-ecosystem-workshops-ame-2019.html)
- If you attend and you provide me with proof of attendance, you can skip the in-class hackathon at 4 PM on Wednesday, and I will give you the extra credit you would have received from the hackathon
- There is a possibility of 5 points extra credit from the hackathon
- The extra credit for the hackathon will be awarded in addition to the 10 point maximum from in class activities and the article review
- You will need a connected PC with Node.JS and Python environments



# Next Steps

- Quiz Extra Credit - Article Review assignment is posted...
- Project 6 active (due 12/11)
- 2<sup>nd</sup> Annual EID Mini-hackathon in class Wed 12/4
  - Working in teams of two, need a network connect dev machine (local or VM) for Python and Node.js
- Next week: Final review and wrap-up, project demos
- New Quiz is up – last one next weekend (class feedback only)
- FCQs are up for a week
- Class staff available to help
  - Shubham - Tues 12-2 PM, Fri 3-5 PM in ECEE 1B24
  - Sharanjeet - Tues 2-3 PM, Thur 2-3 PM in ECEE 1B24
  - Bruce - Tue 9:30-10:30 AM, Thur 1-2 PM in ECOT 242
- Final Exam is set
  - Tuesday Dec 17 7:30 PM - 10 PM ECCR 1B51
  - Final will be open notes and Canvas based, you'll need a PC

