



Class 10-11 CS545

Gregg Vesonder
Stevens Institute of Technology

©2020 Gregg Vesonder

1

Roadmap

- Microinteractions
- Reading: Dan Safer, Microinteractions

©2020 Gregg Vesonder

2

This lecture is all about microinteractions and covers the first section of the book. You need not get the book to understand it. After discussing the project with some of the teams I think understanding and focusing on microinteractions will be helpful. We will also cover more usability patterns in later lectures. This lecture is a bit shorter since you also have the midterm this week!

Four Parts of Microinteraction

- **Trigger** - initiates micro interaction ✓
- **Rules** - determines how it works
- **Feedback** - illuminates the rules
- **Loops and Modes** – meta rules that affect the interaction

Usually begins with an understanding of user needs,
what the user wants to accomplish, **when** they want
to do it and **how** often.

©2020 Gregg Vesonder

So we finished triggers, initiating microinteractions, now we discuss how they work, the rules.

Rules

- **Save As** versus auto save and **Duplicate**
- Opposite of established rules: users have to decide before they make the changes if it should be in a different file.
- One rule to 3 or 4 rules!
- Go Figure and then they do something brilliant like adding an @ sign in a mailer context
- At heart of microinteraction are a set of rules

©2020 Gregg Vesonder

So it begins with one of Apple's rare errors. For those of us that use Microsoft Office we know how convenient save as is. Well apple decide on their preview system that they would ditch save as for a much more cumbersome duplicate and move.

Therefore if you want to save a file under a different name you must first duplicate the file and then move it to the new location – very confusing!

The opposite of such a cumbersome interaction is adding an @ sign to the keyboard when you want the user to provide an email address – I actually consider it to be an egregious user error when a system now asks for an email address without adding an @ sign to the keyboard they provide on a smart phone or tablet.

Rules and the Goal

- Most important part of the rule is the goal of the rule
- It must also be the microinteractions end state.
- Should be focused on the goal (e.g., login) than the steps, the rules
- Rules should be understandable from the goal

©2020 Gregg Vesonder

Rules lead to a goal and all rules must contribute to meeting that goal. A goal for a microinteraction can be enter a password. Why can't all password requests add hints about the composition of the password, e.g., it must contain letters and numbers!

What Rules Determine

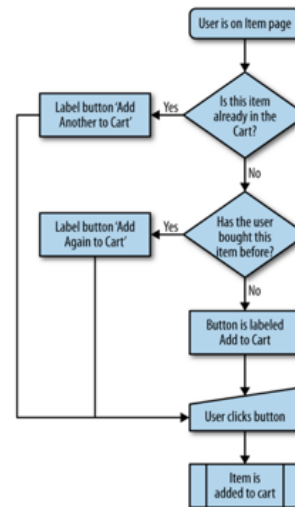
- How the microinteraction responds to the trigger being activated
- What control a user has over the microinteraction in process, e.g., cancel it
- Sequence in which actions take place and their timing
- What data and from where
- What feedback is delivered and when
- What mode the microinteraction is in
- If it repeats and how frequently
- What happens when it ends, if it ends – notification or silently vanish

©2020 Gregg Vesonder

So rules determine what is done – rules in microinteractions usually occur in a sequence and they determine what feedback, if any, is provided to the user.

Constructing rules

- Visualize – flow chart, yes flow chart
- Consider as a sentence – verbs are the action the users can do, nouns are objects enabling the actions
 - Slider raises and lowers volume
 - Every noun should be unique and behave the same in every context
 - Lots of verbs with few nouns



©2020 Gregg Vesonder

In constructing rules it is usually good to draw a flow chart, yes flow chart! I provided an example of the flowchart provided in the book. Note “diamonds” are decision points. For example a rule for a password may permit the user to see the password they are typing by checking a box. Or a rule may turn off the auditory feedback when you are typing your password so no one hears it.

Object states

- Any object can have at least 3 states:
 - Invitation/default state
 - Prepopulated data shown
 - Activated state
 - What does the object do when the user interacts
 - Updated state
 - What happens when the user stops interacting with the object

©2020 Gregg Vesonder

These are very clear. Can you think of any other states, I can think of at least one, inactive state, when an object is grayed out because it is not currently relevant.

Constraints on the rules

- What input and output methods are available?
- What is type and range on any input?
- What is expensive (cpu, radio, battery, disk)?
- What data is available? – sensors, services APIs

©2020 Gregg Vesonder

Constraints are your friend they further narrow the possible interactions with the system and they help you to help the user, which eventually helps you. For example requesting the user to turn off the radio – or automatically turn off the radio to prevent drain of the battery in the smart phone.

Rules should use what is provided

- State of hardware, low battery, brightness (solar)
- What hardware?
- User's past history
- Time of day, noise in the room, user alone
- location

©2020 Gregg Vesonder

The context provides information that can help your app and the microinteraction. What the user has done in the past may help both the user and your app determine what the user would like to do now.

Conservation of Complexity

- A twist on Maeda
- Tesler's Law: "all activities have an inherent complexity; there is a point beyond which you cannot simplify a process any further."
- Either the system handles the complexity or the user handles it
- "Err on the side of removing control and having the microinteraction handle most of the decision making"
 - Do you agree?

©2020 Gregg Vesonder

Sometimes interactions cannot be further simplified. This may provide a dilemma – should the system takeover and help the user make or even make the decision. Being an artificial intelligence researcher, I think the system should – so long as it is reversible!

Limit options, have smart defaults

- The more options provided to the user, the more rules, usually fewer rules result in better and more understandable microinteractions
- Emphasize the most likely next action –
Make it Bigger
 - If you can control where someone is going to look you can control where they are going – Jesse Schell, Art of Game Design

©2020 Gregg Vesonder

So if you are wary of the system deciding, the system can provide hints – for example highlighting the most likely next action (and dimming the impossible or least likely actions).

Controls and user input

- Changing the date and the blinking display
 - Operational simplicity – every command has its own control
 - Perceived simplicity a single control does multiple action (slider, scroll wheel)
- If done repeatedly, perceived simplicity, (except if error is costly)
- Forgiving text fields – phone numbers can be represented in several ways
- Ordering of lists should be carefully designed

©2020 Gregg Vesonder

The first bullet refers to how complex some devices are to change the date. Changing the date is an example of an infrequent interaction and discussing it is really an exercise in how to design microinteractions. Most of the modern devices I have dedicate a separate button to date changing. This has simplified the process. Although it is an infrequent action, not having the correct date is annoying.

Prevent Errors

- Poka-Yoke Principle –“products and processes should be designed so that it is impossible for users to commit an error because the product or process wouldn’t permit it.”
 - E.g, Apple’s Lightning cable

©2020 Gregg Vesonder

From Toyota – this is really a design goal to pursue. One example is Apples lightning cable – it can be plugged in without regard to orientation, unlike classic usb cables!

Four Parts of Microinteraction

- **Trigger** - initiates micro interaction ✓
- **Rules** - determines how it works ✓
- **Feedback** - illuminates the rules ✓
- **Loops and Modes** – meta rules that affect the interaction

Usually begins with an understanding of user needs,
what the user wants to accomplish, **when** they want
to do it and **how** often.

©2020 Gregg Vesonder

So we are now reviewing the third component of the microinteraction, the loops and modes. These activities usually transcend a single interaction.

Feedback

- Slot machines
 - Near wins
 - Great feedback with a win – lights, sound
 - Kaminkow – every action has a unique sound , went from 15 unique sounds to over 400
 - Intermittent reinforcement
- Not!
 - Consistent feedback
 - Constant reinforcement
 - predictable



clipartlord.com

©2020 Gregg Vesonder

Slot machines have a set of meta rules to encourage you to play and insure they turn a profit 😊. They often display near wins to entice the gambler to continue to play and when the user wins not only does the user know it but so does a significant part of the casino crowd. In fact the slot machine has a multitude of sounds to entice the player and bystanders. It also employs sophisticated algorithms to insure that wins are spaced in time to keep the person playing. These schedules were calculated by behavioral psychologists. Slot machines do not have consistent feedback or consistent reinforcement, They are not predictable, and that makes them fun! Seriously.

**Feedback is very powerful
and can make or break
a microinteraction**

©2020 Gregg Vesonder

If you are unsure of this statement consider the total absence of any feedback – without checking external sources such as a process table you would not know whether your app was still operating!

More Feedback

- Feedback illuminates the rules
 - -supports forming a mental model
 - Not necessarily the actual rules
 - But what they can and cannot do with the microinteraction
- **First feedback principle – do not overburden users with feedback**
 - Should be driven by need: what does the user need to know, when and how often

©2020 Gregg Vesonder

Feedback is necessary but you do not want to overburden the user with feedback – too much feedback overwhelms users and they often leave.

Feedback *should* occur

- Immediately after a manual trigger or following/during the manual adjustment of a rule
 - All user-initiated actions should be accompanied by a system acknowledgement
- On any system initiated triggers in which the state of the interaction has changed significantly
- Whenever the user reaches the edge of a rule
 - Likely an error is about to occur
- Whenever the system cannot execute a command
 - But don't be hasty – try a few times, connecting to a network for example
- Show progress on any critical process the will take a long time

©2020 Gregg Vesonder

These are instances when feedback should occur. Any action that you do should be acknowledged by the system – not doing so will result in a user unsure of whether the system is still running. Similarly when something takes a long time the system should update the user on progress and insure that the application is still alive.

Feedback *could* occur

- At the beginning or end of a process
- At the beginning or end of a mode or when switching between modes

©2020 Gregg Vesonder

This is optional – feedback is necessary when you are in a moded text editor – one that has either an input or a command mode – vi or vim is a great example of this.

Feedback is for Humans

- Something has happened
- You did something
- A process has started/ended
- A process is ongoing
- You can't do that!
- Text is not always an option
 - No screen real estate
 - A substantial portion of the population is illiterate 793 million adults
- The penalty for acting on feedback that may be misinterpreted should be none
- Errors improve performance
- Feedback can convey personality – long download ("grab a snickers")
 - beware being annoying or overdoing

©2020 Gregg Vesonder

This reflects Norman's comment in the preface of the book. He stated that an error message is not that useful if it does not provide a remedy.. But there are challenges to feedback especially on the relatively small screens of smartphones and, if a basic services app (medical care, food, ...) you must insure it can be understood by folks with limited education – iconics and pictures are helpful. Feedback also can have personality in moderation. Google often does a great job of that.

Feedback Principles

- Do not overburden users with feedback
- The best feedback is never arbitrary
 - Deep connection between action and feedback
- Use the least amount of feedback you can to convey a message – but test!
- Use the overlooked as a means of message delivery (e.g. cursors)

```
downreader - ssh gregg@danube.stevens.edu - 80-24
Last login: Sat Nov 21 20:44:27 on ttys002
Belle:~ downreader$ ssh gregg@danube.stevens.edu
Password:
```

©2020 Gregg Vesonder

Use feedback judiciously – too much will be ignored and clutter your application. Using cursors for feedback is a great idea – for example when a password is asked on the command line screen of OS X (basically linux) the cursor changes to a key icon. And of course test the appropriateness and effectiveness of your feedback with users. Hopefully that statement does not surprise you! 😊

Feedback Methods

- Visual – majority of feedback (remember accessibility!)
 - Don't show redundant feedback
 - The more frequent the feedback, the less intrusive
 - No blinking unless you must pay attention to it!
- Animation
 - movement is powerful, but use sparingly and try to avoid it
 - It can be powerful if it accurately conveys how the microinteraction works
- Messages – be careful that it is accurate
 - Should be measured in words not sentences
- Audio – used sparingly, e.g., emphasis and alerts
 - Foghorn test
- Speech
- Haptics

©2020 Gregg Vesonder

Of course feedback must take into account accessibility – almost all visual feedback should have a text version so that it can be translated to text to speech for visually impaired individuals. Feedback should be crisp measured in number of words rather than number of sentences. Audio is necessary for GPS, e.g., turn left, but on smart phones often the sound is muted, is there alternative feedback? Haptics are very useful – at the very least they alert the user that the system needs your attention.

Reasons for using animation

- Maintaining context while changing views (scrolling a list)
- Explaining what just happened (poof of smoke on delete)
- Showing relationships between object (trash can and file)
- Focusing attention – alerting that an item (price) changed value
- Improving perceived performance (progress bars)
- Creating illusion of virtual space – transitions for modes
- Encouraging deeper engagement – animations invite interaction

©2020 Gregg Vesonder

In the early days of the web animation was abused – try this website and be sure to enable sound : <http://www.angelfire.com/super/badwebs/> . That web site should drive home many of the points made here and on the next page.

Animations should be(google)

- Fast
- Smooth
- Natural
- Simple
- Purposeful – not eye candy

©2020 Gregg Vesonder

Google has provided a great set of advice for animations. My caution is make sure you really, really need the animation before doing it!

More audio - earcons

- Icons for sound – a ping versus you have mail
- abstract so should be carefully chosen
- Avoid shrill earcons
- Should be short except if you are indicating a long process – then background noise
- Any earcon in a loop will probably become annoying!

©2020 Gregg Vesonder

The website I provided a few slides back (slide 24) had obnoxious looping sound – please do not even think of doing that! Most systems and smart phones use a variety of earcons to indicate incoming mail, outgoing mail, new SMS messages, Often they are customizable and you have the option to disable them. For average apps always have the option of disabling sound notifications and sound in general.

More audio - speech

- Brief and clear (a trend, not the micro in microinteractions)
- Word prompt at the **end** of the message
 - “To turn sound off say, yes” instead of “say yes to turn sound off”
- Use recorded messages versus Text-to-speech (still a bit inhuman)

©2020 Gregg Vesonder

I am still not a fan of multiple sentences of generated audio. Note that, unlike pictures or text, speech is serial – you cannot hear the whole sentence said but only one word at a time. For that reason, even though it may result in inferior grammar, try to place the prompt choices at the end of the speech. Also a technical bit – text to speech is the software stitching together words into an entire sentence. Often this sounds a bit odd and becomes annoying for more than a sentence or so. If you can survive on stock phrases have a person with a great voice – not mine – record them and play back when appropriate.

Haptics

- New with advent of pagers, mobile phones and game controllers
- Device must be in close proximity of user
- Face and hands most sensitive, legs and torso not so much
- Not a dense information medium – complex messages are difficult to convey
- use it to:
 - Enhance physical action
 - As an alert when audio is not available or undesirable
 - Create artificial texture or friction

©2020 Gregg Vesonder

A major way my apple watch communicates with me is through haptics – it vibrates to get my attention – very handy. Since it is on my hand I am sensitive to the vibration, there are times that I do not even sense the vibration of phones in my pocket. Besides alerts it is especially effective in gaming and draws you further into the virtual reality.

More feedback rules

- **Contextual changes**
 - if night time does volume increase or decrease?
 - If night time does illumination increase or decrease?
 - Are both also a function of battery life?
- **Duration**
 - How long, what dismisses it
- **Intensity**
 - How bright, loud, vibrating is the effect?
- **Repetition**
 - Is it repeated, how often, how long

©2020 Gregg Vesonder

All of these really should be user tested and even then, user adjustable. On any electronic device one has to be mindful of battery life too!

Four Parts of Microinteraction

- **Trigger** - initiates micro interaction ✓
- **Rules** - determines how it works ✓
- **Feedback** - illuminates the rules ✓
- **Loops and Modes** – meta rules that affect the interaction

Usually begins with an understanding of user needs,
what the user wants to accomplish, **when** they want
to do it and **how** often.

©2020 Gregg Vesonder

So we reviewed feedback and we are onto loops and modes that provide continuity to user interactions.

Modes

- Mode is a fork in the rules
 - Best reason to have it is for an infrequent interaction that would clutter microinteractions main purpose, e.g., settings
 - But modes are error prone, they are often near invisible
 - Fewer modes, zero is best,
 - Provide the mode with its own screen – blatantly signal it
 - Emphasize transitions between modes

©2020 Gregg Vesonder

So a mode is a change of context. For example – a set of screens that provides settings for the application, e.g., volume, sound effects, control choice (keyboard arrows or mouse), Mode also can be used to differentiate between commands and text entry for example vim, a text editor. It is very important to do something special to clearly indicate which mode. Digital watches are infamous for providing poor indicators of mode.

Although many folks encourage avoiding modes, they can be very helpful if done correctly.

Modes 2

- Spring loaded and “one of” modes
 - Spring loaded – pressing a key, as soon as action stops so does the mode, e.g., shift key
 - User seldom forgets since they are doing something physical to make it possible
 - “one of” – also not a problem usually since the user initiates a mode that lasts one action, e.g., highlight
 - useful in voice interactions (alexa play music)

©2020 Gregg Vesonder

Note that often the shift key provides problems for passwords so even such a commonly used mode is error prone, especially if the spring loaded locks. Mode also is useful for specific actions and to protect privacy – for example using the amazon echo – the echo only listens when you use the word alexa.

Loops

- A cycle that repeats usually for a set duration
- A loop is usually indicated by rules either directly or indirectly

©2020 Gregg Vesonder

Loops are different it is an interval of time where actions are remembered or alerts set or Very analagous to typical program constructs such as *while* or *for* and may even use them in the implementation.

Styles of loops

- Count controlled loop (for) – do it a set number of times
- Condition controlled loop (while)
- Collection controlled loop – runs through everything in a set
- Infinite loop

©2020 Gregg Vesonder

In particular an infinite loop `while (true) do {}` is great to begin understanding the user and adapting to the user behavior. You record what the user does and try to understand it perhaps readjusting menus in the applications once you understand what the user typically does.

Kinds of loops

- Open loops do not respond to feedback
- Closed loops have a feedback mechanism embedded and are self adjusting
 - As engine noise increases so does volume on stereo
- Long loops - closed loops that adapt to behavioral feedback

©2020 Gregg Vesonder

An example of an open loop are setup like commands, e.g., every day at 7pm turn on the driveway lights or turn lights on at sunset and off at sunrise. Long loops provide the ability to adjust the menus in the example I provided on the previous slide. This is often where artificial intelligence is used.

Loop lore

- Can be used to make sure an action does not go on too long – banking apps logs you off after a few minutes of inactivity
- Loops can be used to prompt if no action occurs

©2020 Gregg Vesonder

Loops also can be protective. They can notice inactivity and log you off in a reasonable time or prompt you that an input is desired and you have not provided it or let you know that a task in a game is done. The uses are endless.

Long loop

- Closed loop that adapts over time due to behavioral feedback
- Long Wow – adapts microinteraction so that it feels customized or even brand new
- Extend microinteraction far beyond a single instance of use
- E.g., items placed on a wish list or shopping cart. Consistently playing music loud alters scale.

©2020 Gregg Vesonder

The long wow is being surprised occasionally when the long loop alters the application environment to be sensitive to user needs – for example reordering the menu reflecting how the user is applying menu items. Other examples are reminding you about items placed in the shopping cart that you have not purchased. Another example is adjusting controls and scales (it is easier to get to loud) according to your preferences.

More Long Loop

- Progressive reduction where microinteraction becomes simpler, less guidance.
 - But if the user has not used in a while it reverts to more guidance
- Progressive disclosure – shortcuts added or feature revealed



©2020 Gregg Vesonder

The application with a long loop can become sensitive to use – less guidance/help is offered when you have used it consistently and more guidance and help after a long lapse, e.g., in Angry Birds reminding the user what each bird's function is.

Four Parts of Microinteraction

- **Trigger** - initiates micro interaction ✓
- **Rules** - determines how it works ✓
- **Feedback** - illuminates the rules ✓
- **Loops and Modes** – meta rules that affect the interaction ✓

Usually begins with an understanding of user needs,
what the user wants to accomplish, **when** they want
to do it and **how** often.

©2020 Gregg Vesonder

So we finished a whirlwind tour of the components of micro interaction. Let's explore some examples and how to design microinteractions more effectively.

Microinteractions- Examples

- Mobile alarm App
- Goal – to be alerted
- Rules:
 - User selects alarm time
 - Alarm goes off at that time
 - User turns off alarm
- Bring the data forward (not easy)
 - How many alarms set in badge



<http://www.bigcheesebadges.com/number-2-stars-25mm-pin-button-badge-p-1600.html>

©2020 Gregg Vesonder

This is very straightforward. Badges can be used to signify the number of alarms. This badge I lifted from bigcheesebadges could signify that 2 alarms are set.

Alarm App Continued

- Don't start from zero
 - What do you know?
 - Use it!
 - If alarm set more than 3 days in a row and not set on 4th prompt
 - Use format of country
 - Display all set alarms and time until it sounds
- Optimize for what folks do most of the time
- The alarm itself – novel and when you turn it off, a nice solid click

©2020 Gregg Vesonder

The “if alarm set more than 3 days and suddenly not set” is an example of another loop. Of course use the numeric format indigenous to the country. Buttons providing feedbacks – clicks are very satisfying just like a well made laptop clip.

Next – Online Shared Play List

- Goal – discover and share new music
- Rules:
 - New song arrives add to playlist
 - User can add song to playlist
 - New songs added at top of playlist
- Triggers:
 - Adding a song (manual trigger)
 - Friend adding a song remotely (system trigger)

©2020 Gregg Vesonder

Microinteractions are everywhere! Note that the really interesting microinteractions are deeply intertwined with the application, knowledge of what the application is doing, the domain, is very important

Prototyping and Documenting Microinteractions

- Prototype on the host platform
- Make a movie
- Create frame-by-frame storyboards
- **NOT static screen shots**

©2020 Gregg Vesonder

Microinteractions are by their very nature interactive and “alive.” A static button does not make it – have it click and do something.

Fixing Microinteractions

- Should this be a signature moment – memorable, richer (custom controls) feedback
- Are you starting from zero? What do you know about the user and the situation
- What is the most important piece of data (goal) and can I bring it forward
- Would a custom control be appropriate? Salience!
- Am I preventing human errors
- Am I using what is overlooked (e.g. cursor)
- Are the text and icons humane – can humans relate?
- Add animations to make it less static
- Additional redundant channels of feedback
- What is the long loop? When user returns 2nd or 100th time

©2020 Gregg Vesonder

As is typical in many User Experience discussion there is always a set of suggestions, since there are few hard and fast rules.

It is important to know your user and the tasks they are attempting. The most important goal should be highlighted – for example in Angry Birds it is playing the game not buying merchandise!

Much of it requires a user centered approach, what does the user population understand and what knowledge should you begin collecting about them and the application and how can you use it to make interacting with the application even more enjoyable. This knowledge is really located in the long loop and you should treat the user differently early in their experience than you will later when you can hopefully streamline the experience for them.



Think Small

- It is the tiny moments, the microinteractions, that can make large systems humane
- Think small and change the world

©2020 Gregg Vesonder

This summarizes the philosophy of the book, but wait, there is a bit more.

During testing

- Ensure that you truly understand the goal of the microinteraction and that it is not just a step in the process
- Understand what data is important
- Is microcopy necessary and is it clear and understood, especially labels
- By observation:
 - Are there too many clicks/taps/controls/presses?
 - Any confusion as to why the user is doing something?
 - Any confusion from the user on what just happened
 - Did anything happen?
 - I can't find what I am looking for
 - I don't know where I am
 - If I click/push/tap this what happens – user is confused – need microcopy?
 - I did not know I could do that – hidden action (eg., multitouch) camera zoom
 - What do I do know?
 - What am I seeing?

©2020 Gregg Vesonder

Again microcopy is the labels on buttons or instructions. This slide really is a heuristic evaluation geared towards microinteractions. Is it too complex? Is the microcopy irrelevant or too wordy?

Everything should be visible, users should not be surprised to find a feature.

Quantitative Data

- Completion rate
- Overall duration of microinteraction
- Duration of specific steps
- Number of steps
- Number of clicks/taps/selects
- Number of system errors
- Number of human errors
- Users can rate:
 - Satisfaction
 - Difficulty
 - Confidence
 - usefulness

©2020 Gregg Vesonder

So what should you collect – this is a great list for evaluating E's! Note all except the last are very objective, countable things – the last sub list, what users can rate, provides you with a flavor of how the users feel about the application at a higher level, do they like it? Is it too hard? Too easy? Is it useful? All of those ratings shed light on these questions but are best considered in conjunction with more objective data.

I hope you enjoyed your tour of microinteractions and will design and use them in future endeavors.