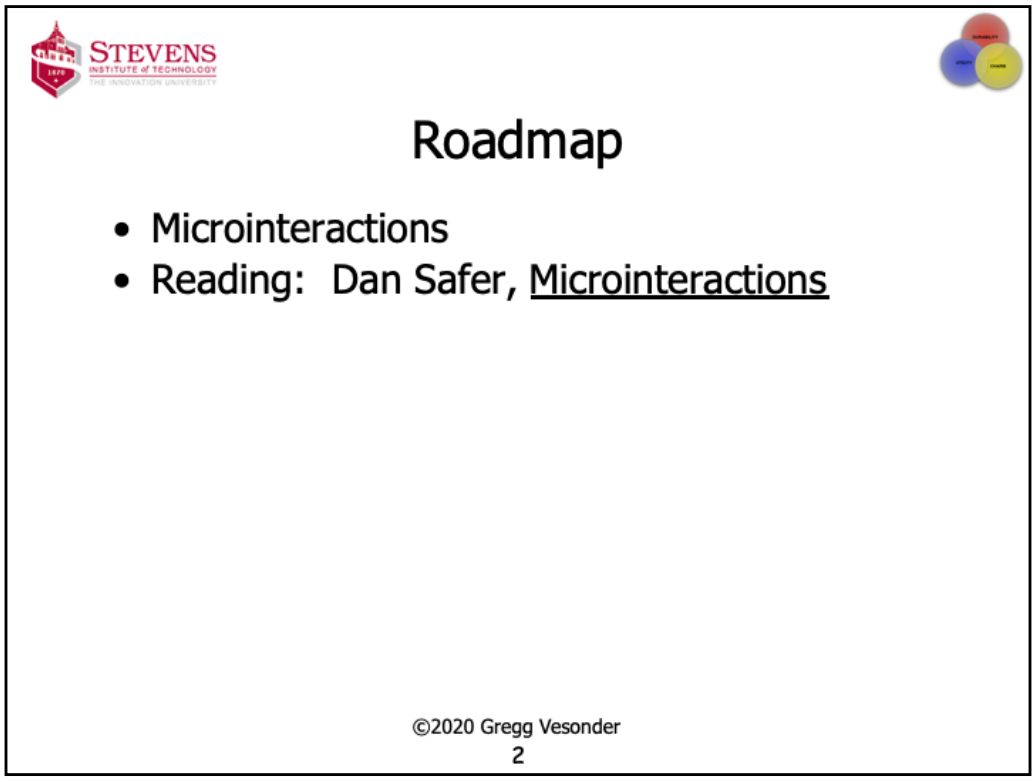
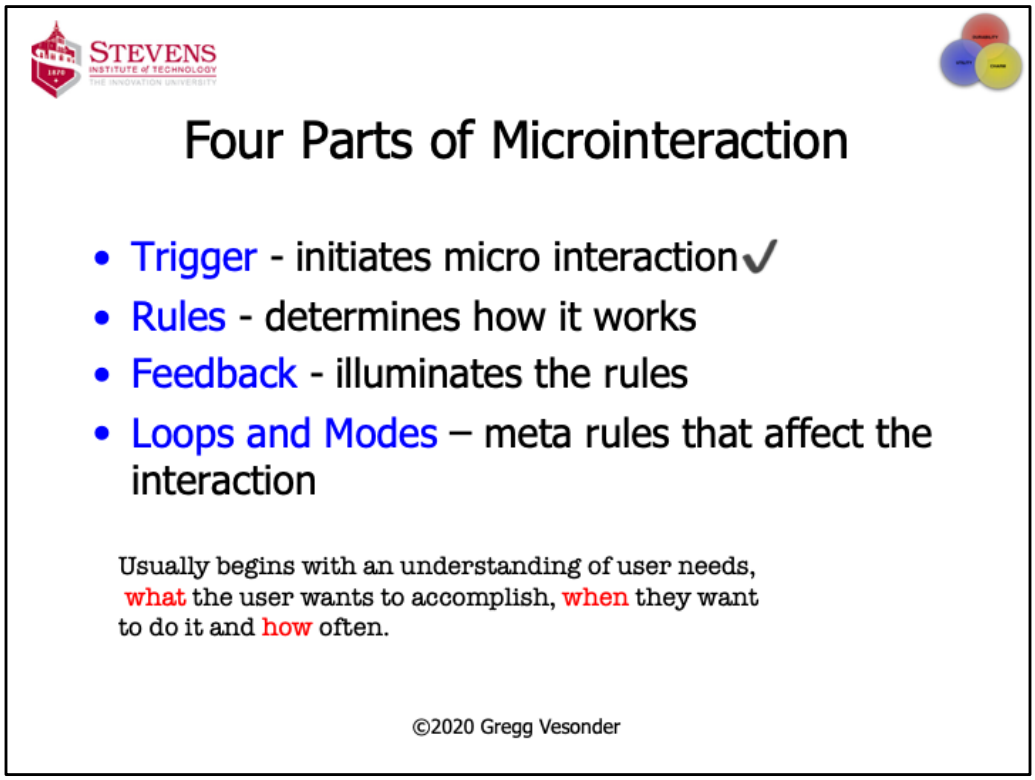
Class 10-11 CS545

Gregg Vesonder
Stevens Institute of Technology

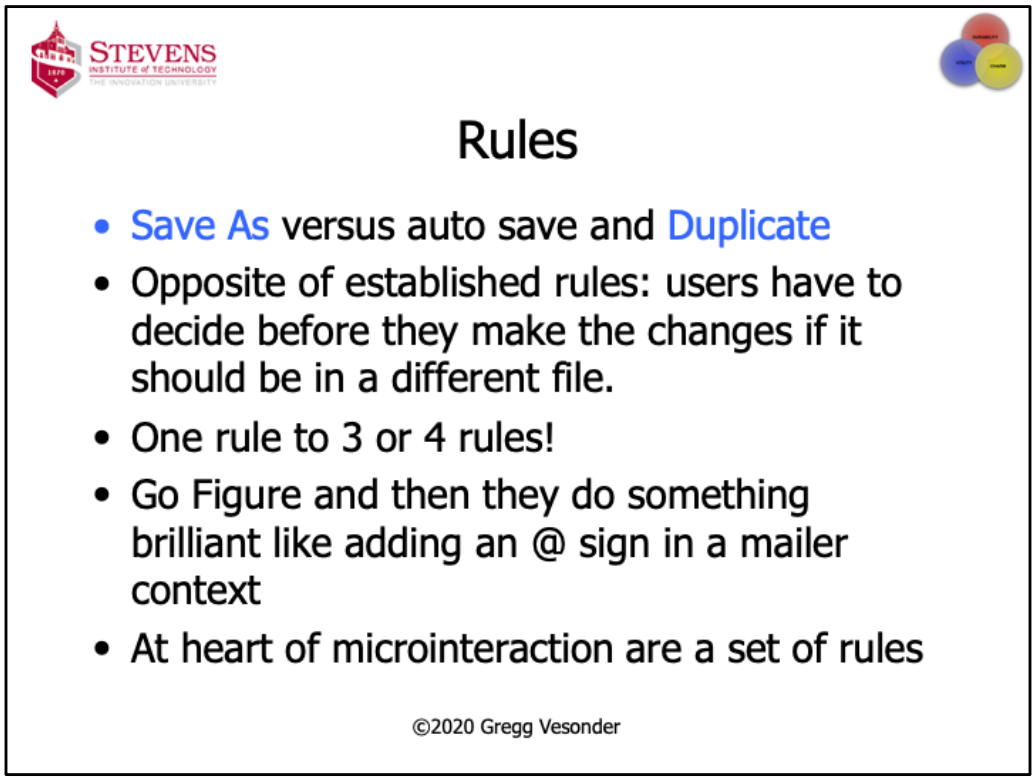
1



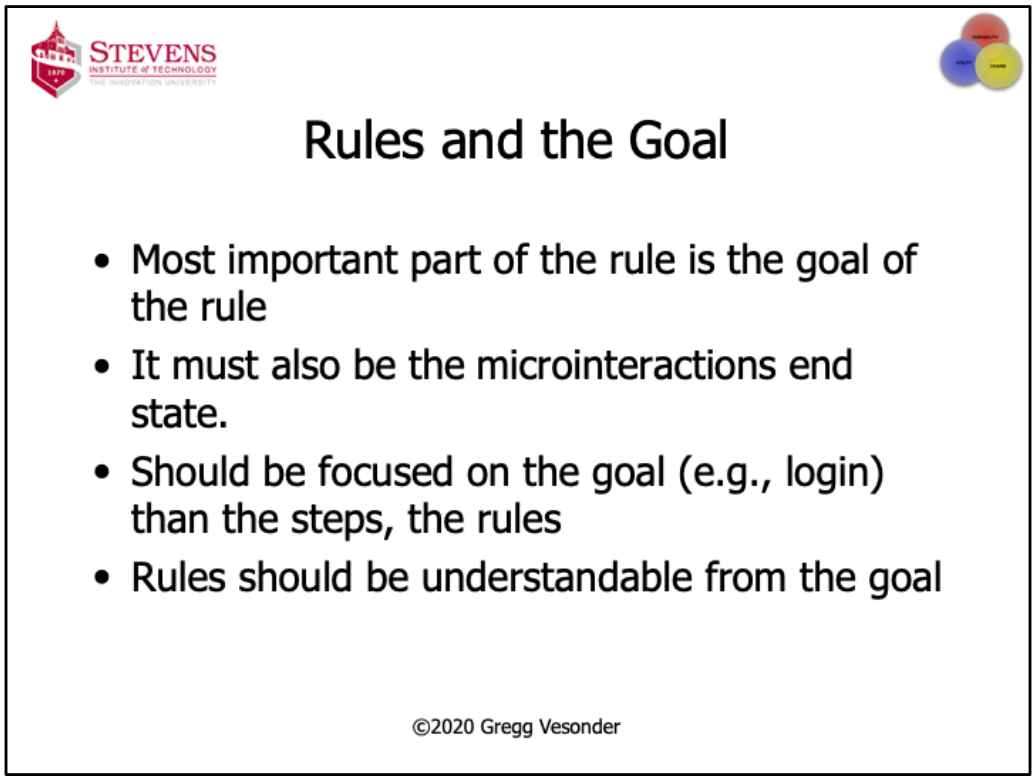
This lecture is all about microinteractions and covers the first section of the book. You need not get the book to understand it. After discussing the project with some of the teams I think understanding and focusing on microinteractions will be helpful. We will also cover more usability patterns in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



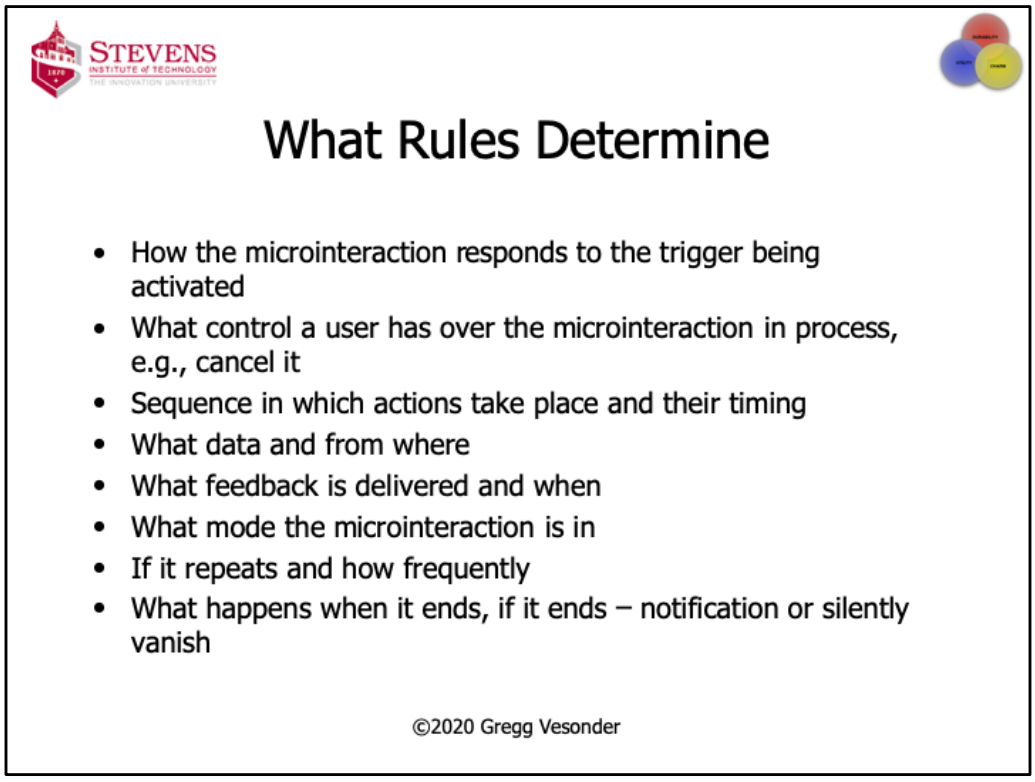
So we finished triggers, initiating microinteractions, now we discuss how they work, the rules.



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.



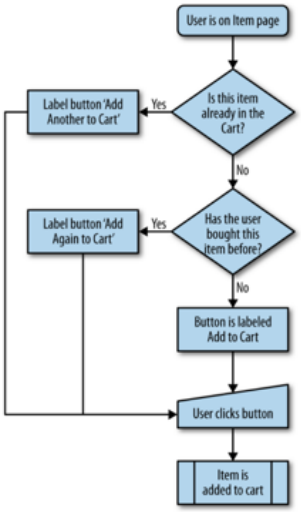
Rules lead to a goal and all rules must contribute to meeting that goal. A goal for a microinteraction can be enter a password. Why can't all password requests add hints about the composition of the password, e.g., it must contain letters and numbers!



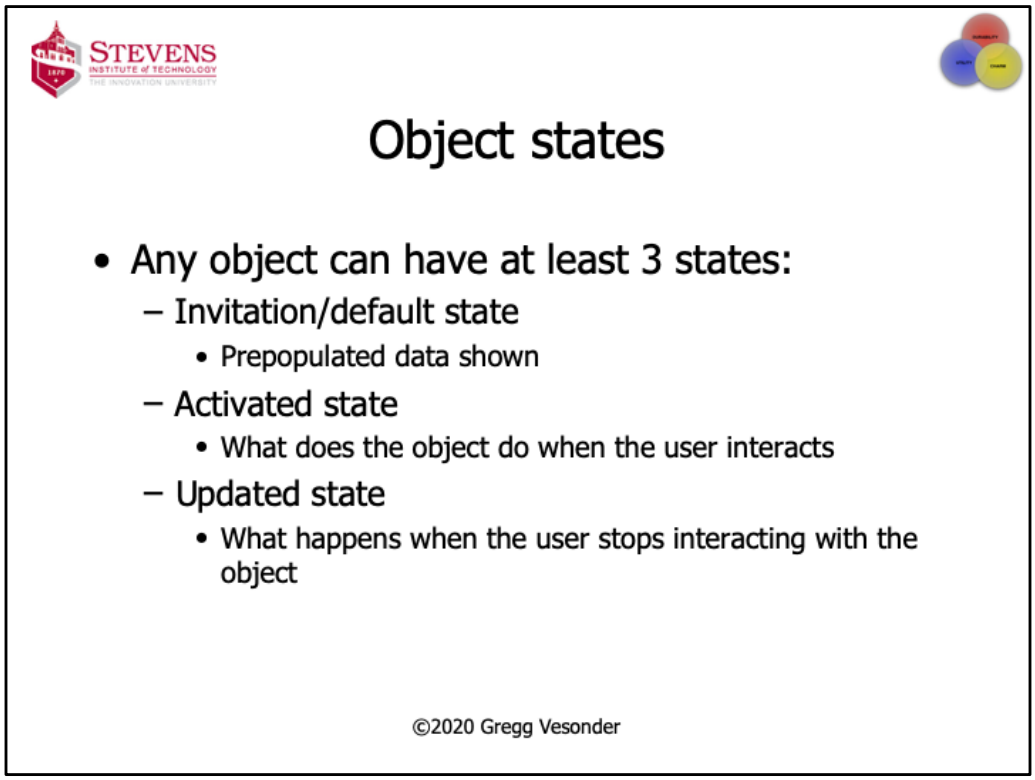
So rules determine what is done – rules in microinteractions usually occur in a sequence and they determine what feedback, if any, is provided to the user.

Constructing rules

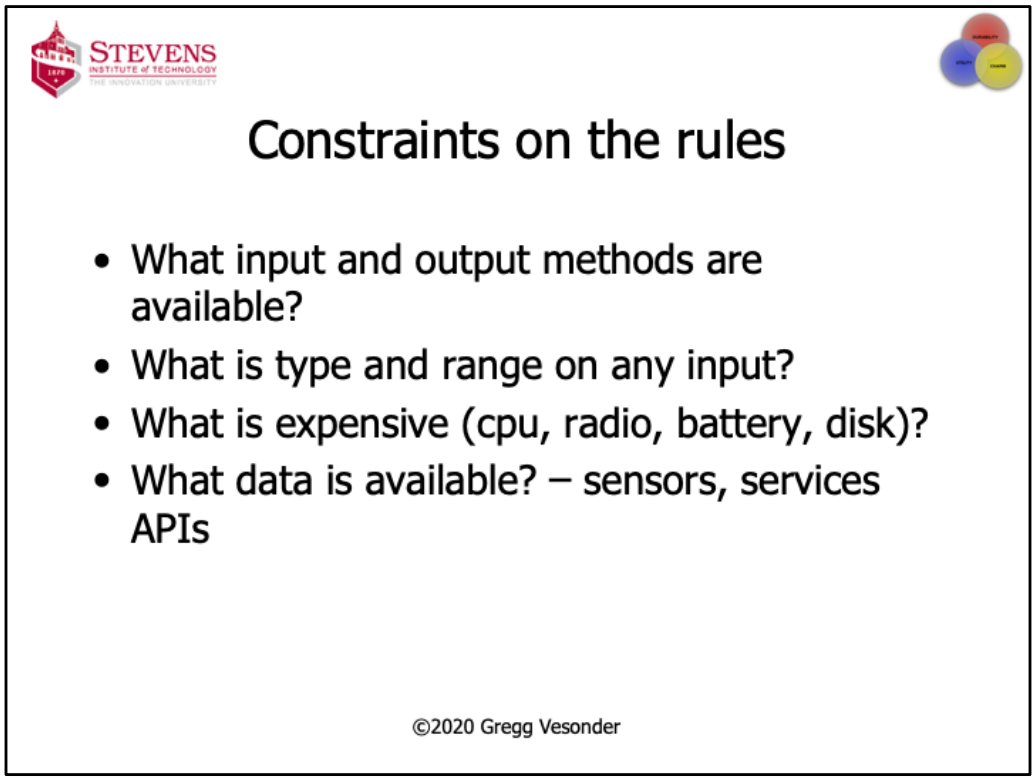
- Visualize – flow chart, yes flow chart
- Consider as a sentence – verbs are the action the users can do, nouns are objects enabling the actions
  - Slider raises and lowers volume
  - Every noun should be unique and behave the same in every context
  - Lots of verbs with few nouns



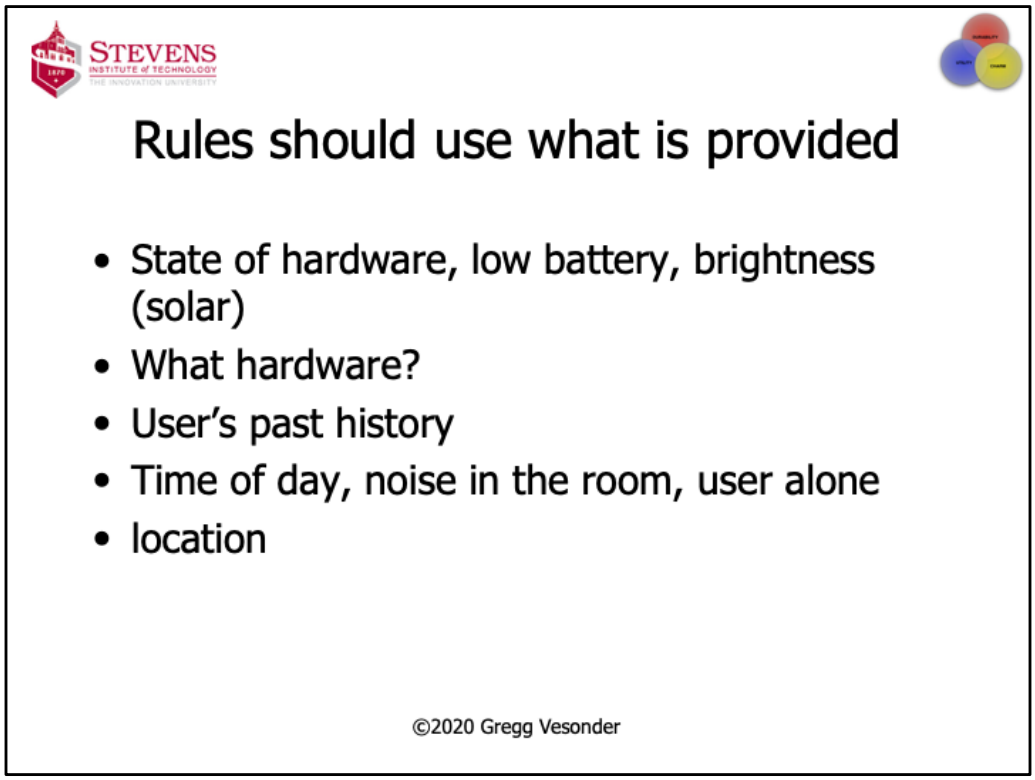
In constructing rules it is usually good to draw a flow chart, yes flow chart! I provided an example of the flowchart provided in the book. Note "diamonds" are decision points. For example a rule for a password may permit the user to see the password they are typing by checking a box. Or a rule may turn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



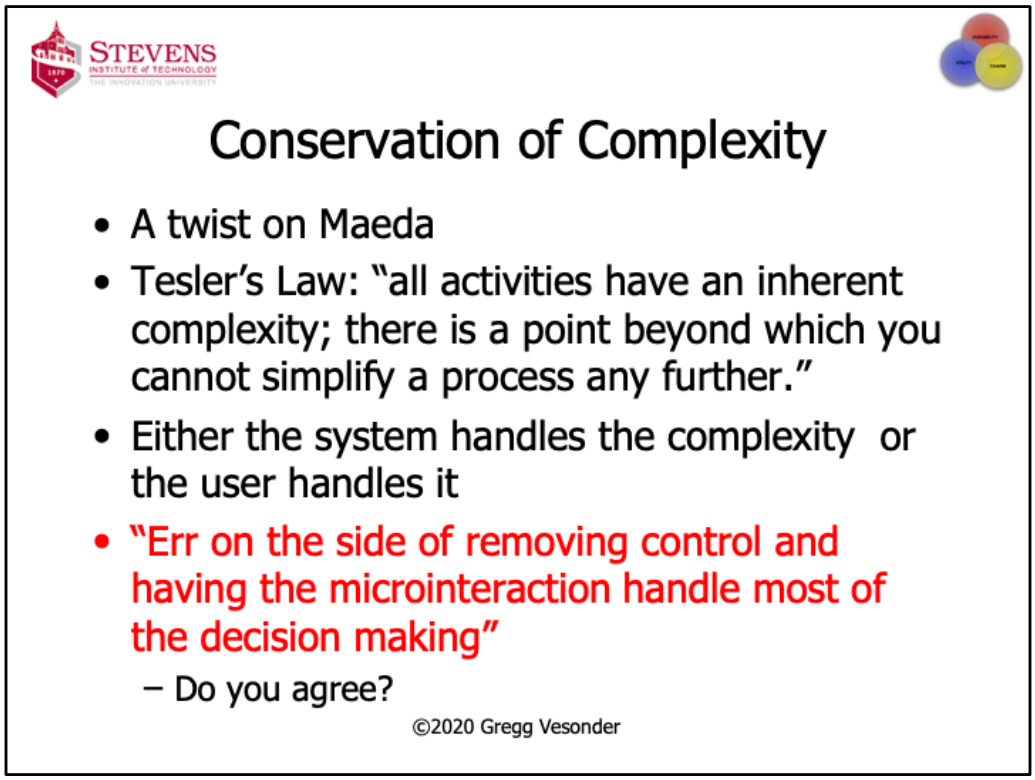
These are very clear. Can you think of any other states, I can think of at least one, inactive state, when an object is grayed out because it is not currently relevant.



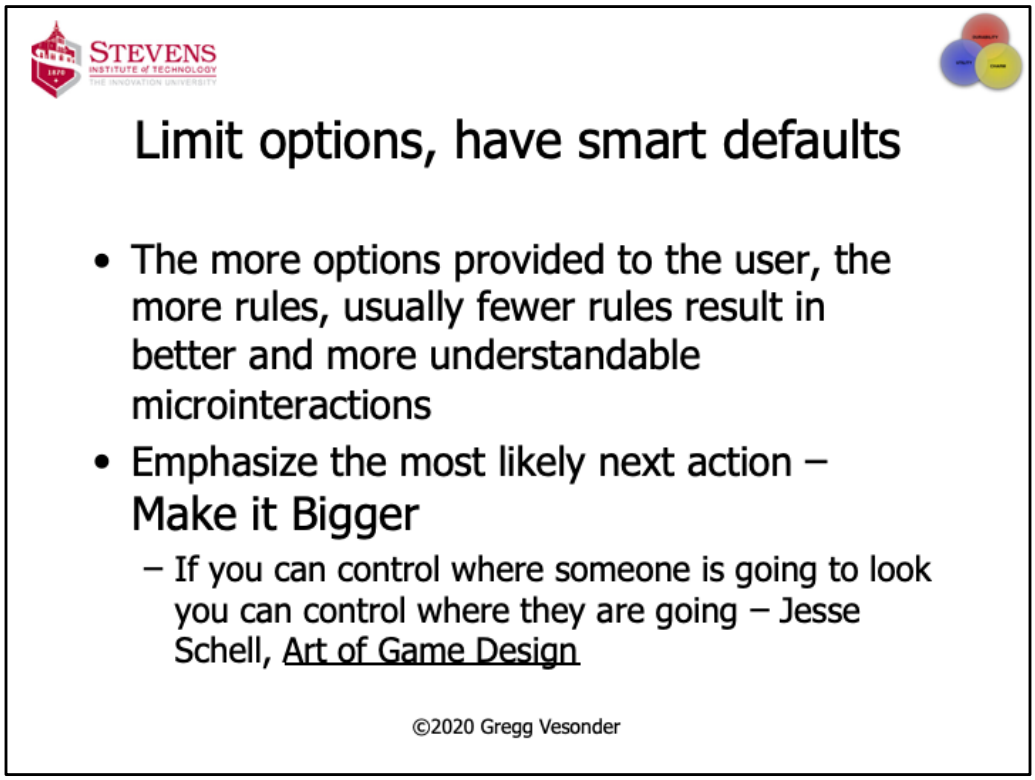
Constraints are your friend they further narrow the possible interactions with the system and they help you to help the user, which eventually helps you. For example requesting the user to turn off the radio – or automatically turn off the radio to prevent drain of the battery in the smart phone.



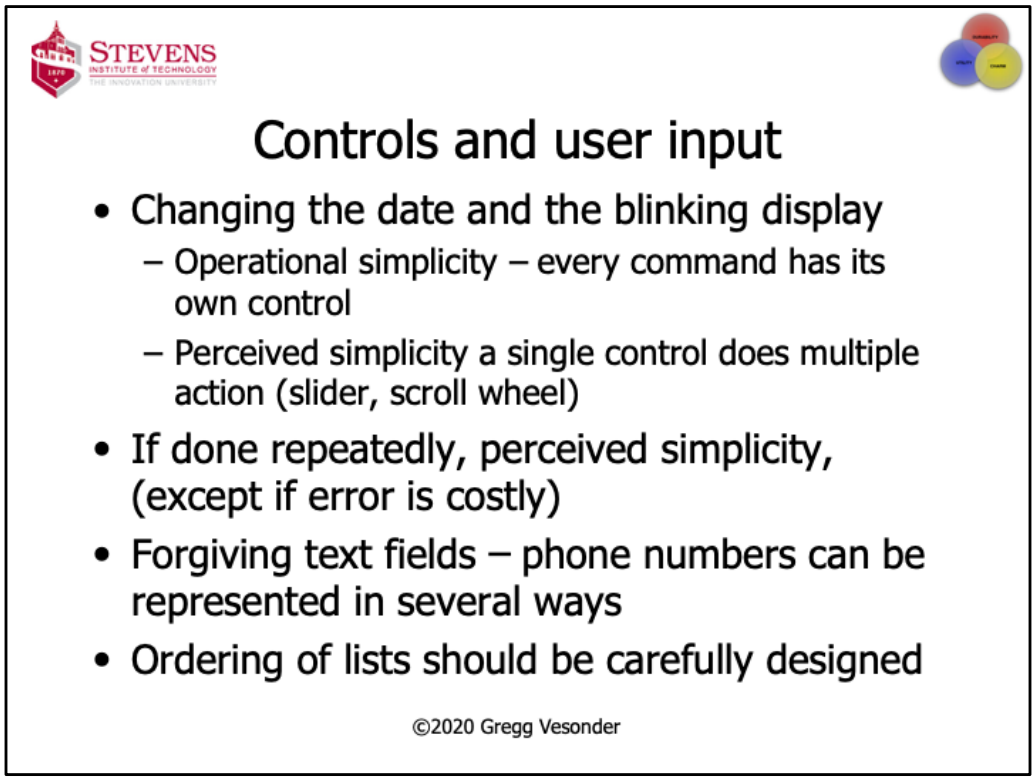
The context provides information that can help your app and the microinteraction. What the user has done in the past may help both the user and your app determine what the user would like to do now.



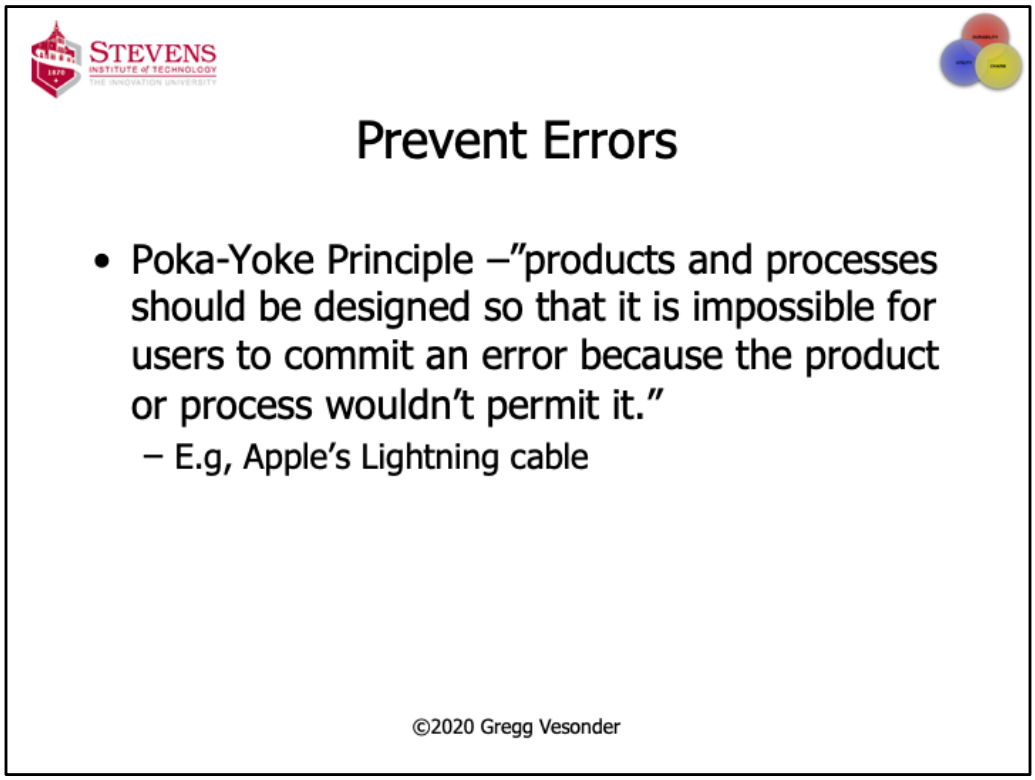
Sometimes interactions cannot be further simplified. This may provide a dilemma – should the system takeover and help the user make or even make the decision. Being an artificial intelligence researcher, I think the system should – so long as it is reversible!



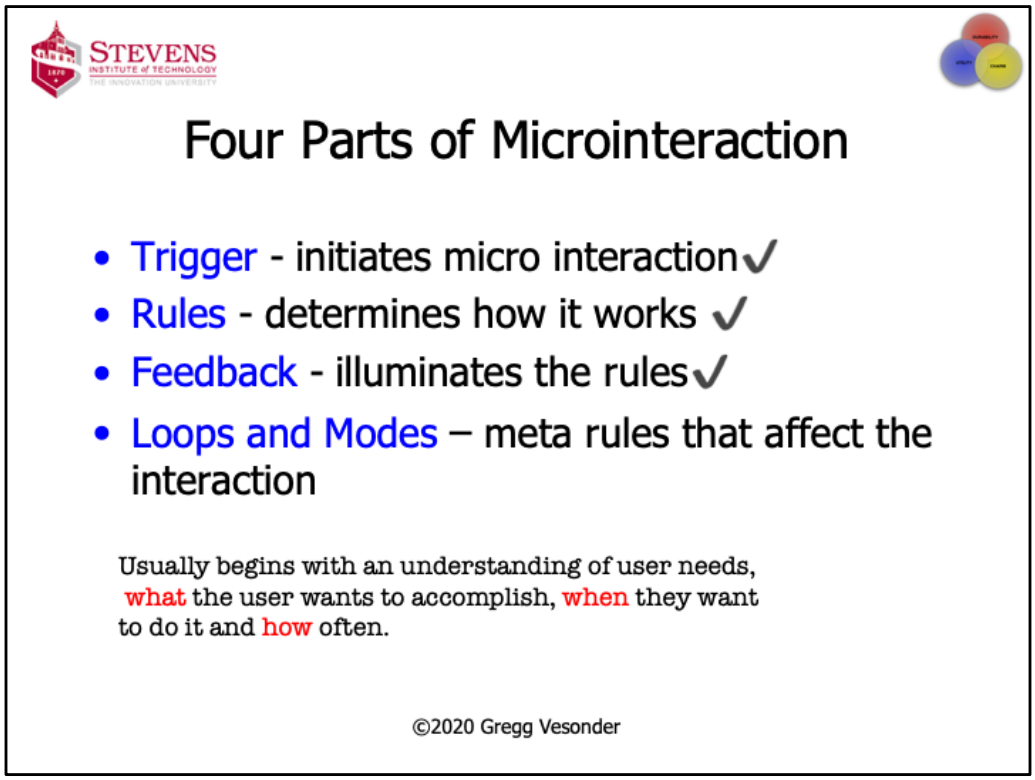
So if you are wary of the system deciding, the system can provide hints – for example highlighting the most likely next action (and dimming the impossible or least likely actions).



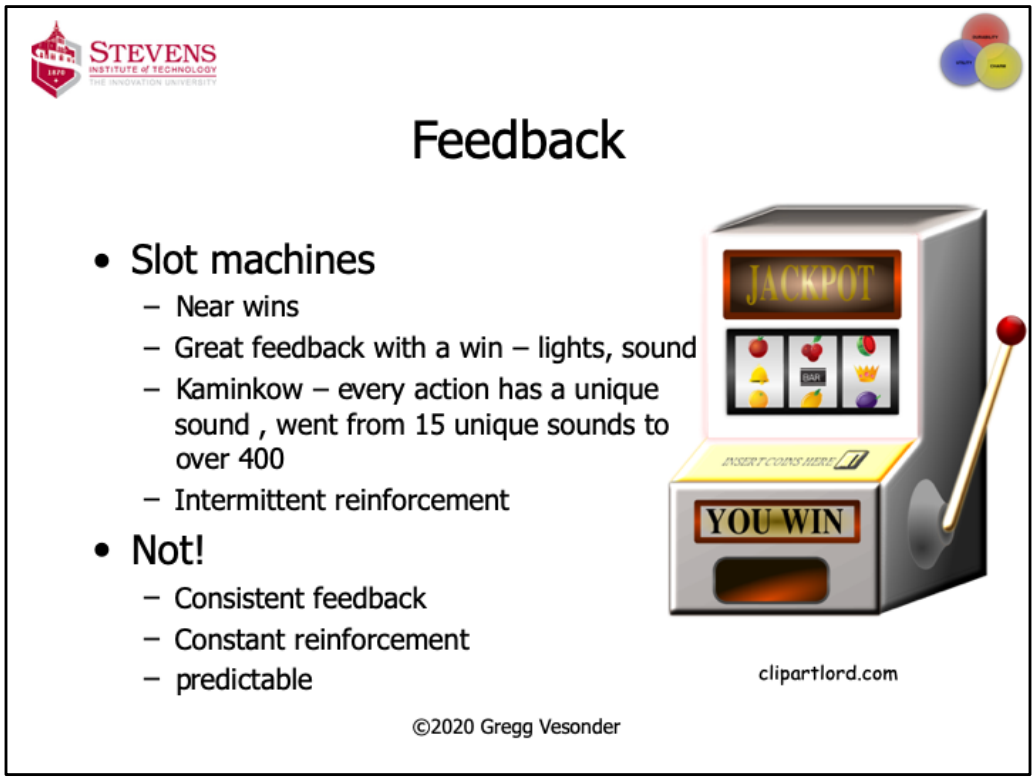
The first bullet refers to how complex some devices are to change the date. Changing the date is an example of an infrequent interaction and discussing it is really an exercise in how to design microinteractions. Most of the modern devices I have dedicate a separate button to date changing. This has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



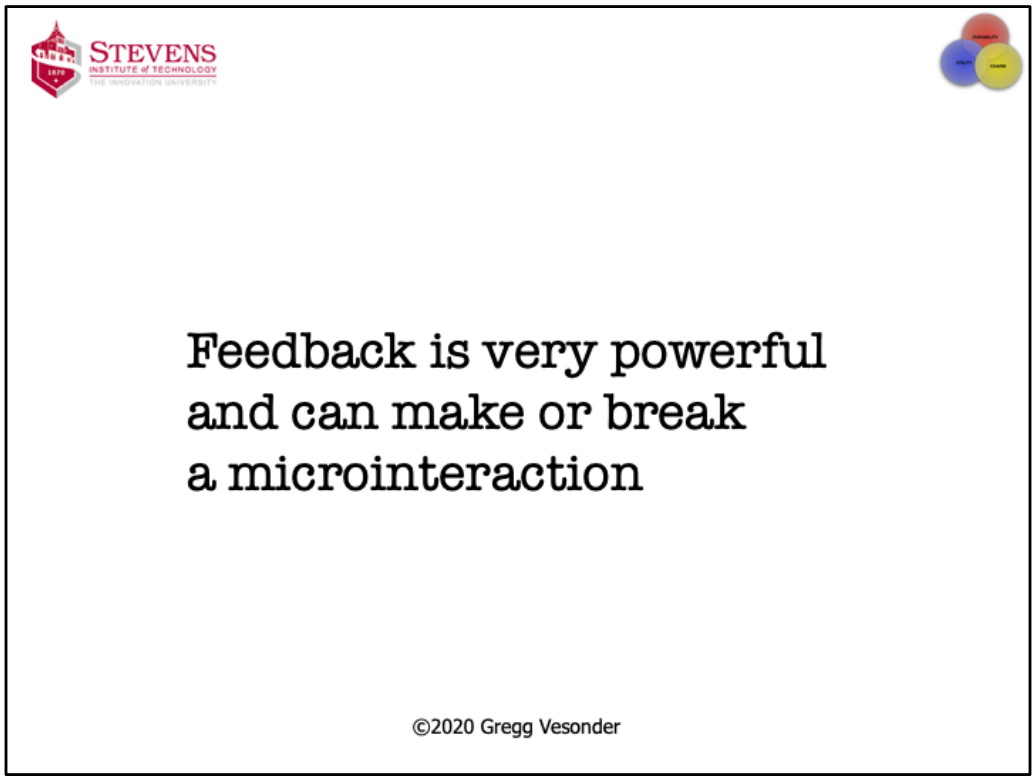
From Toyota – this is really a design goal to pursue. One example is Apples lightning cable – it can be plugged in without regard to orientation, unlike classic usb cables!



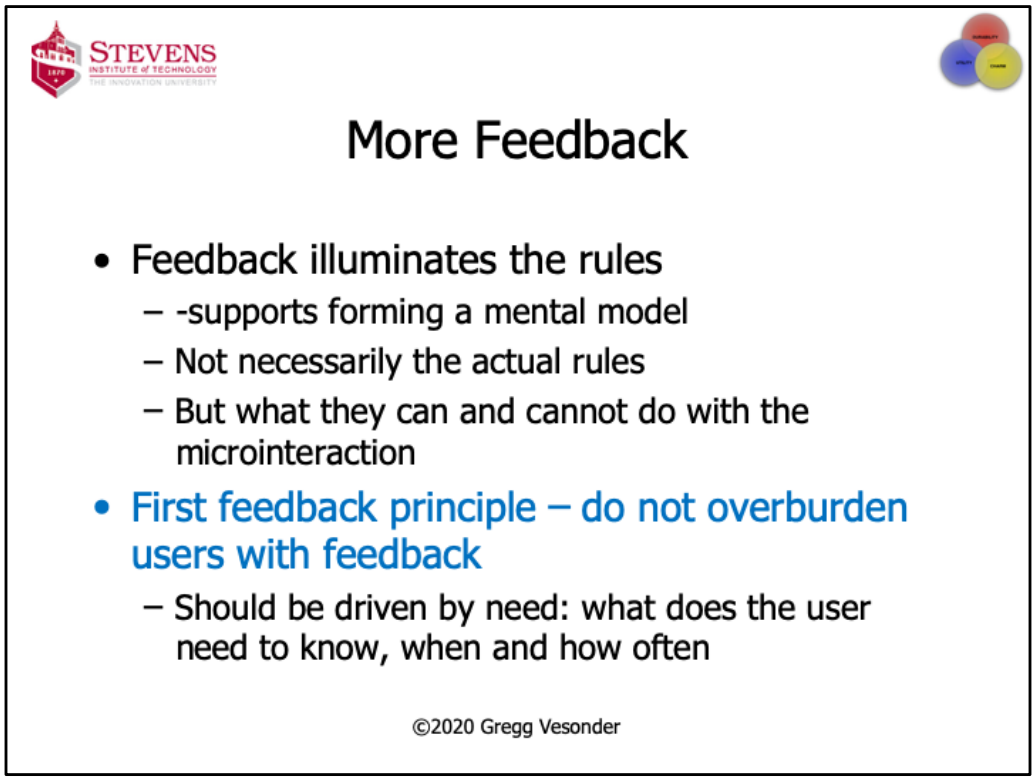
So we are now reviewing the third component of the microinteraction, the loops and modes. These activities usually transcend a single interaction.



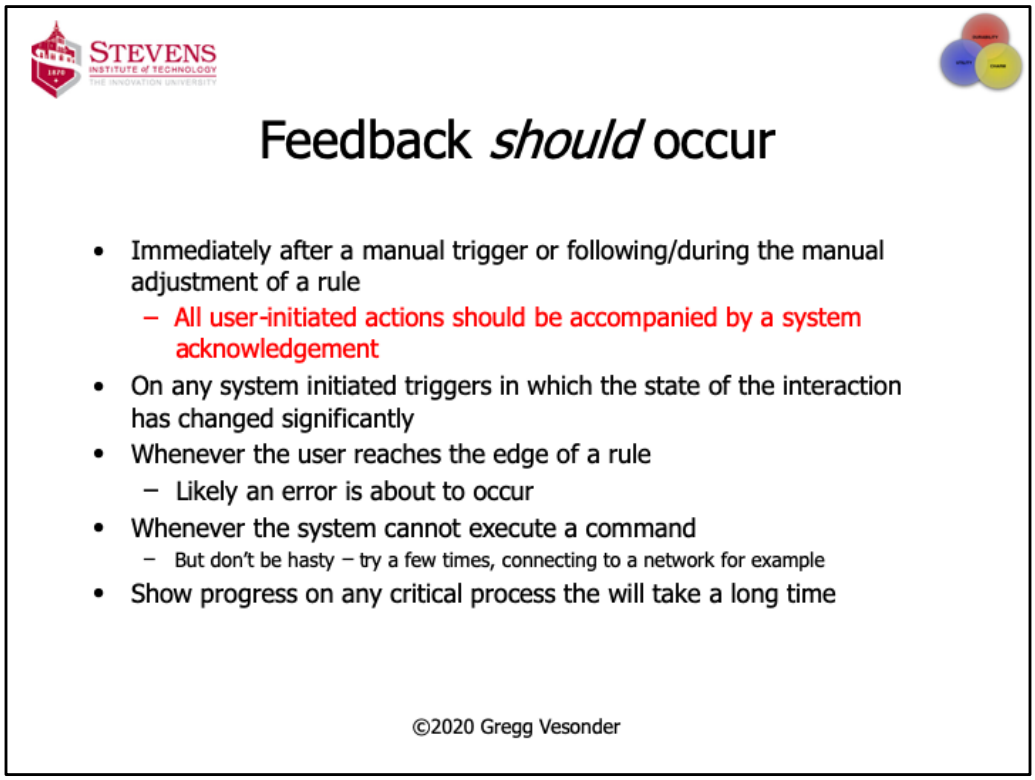
Slot machines have a set of meta rules to encourage you to play and insure they turn a profit ☺. They often display near wins to entice the gambler to continue to play and when the user wins not only does the user know it but so does a significant part of the casino crowd. In fact the slot machine has 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.



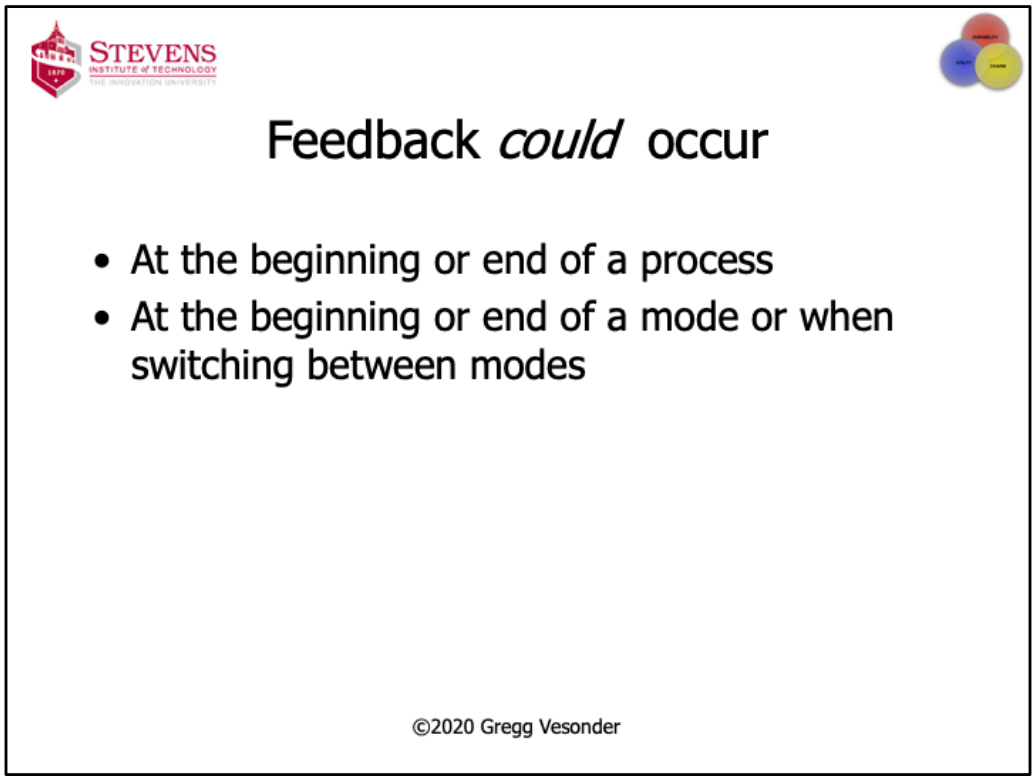
If you are unsure of this statement consider the total absence of any feedback – without checking external sources such as a process table you would not know whether your app was still operating!



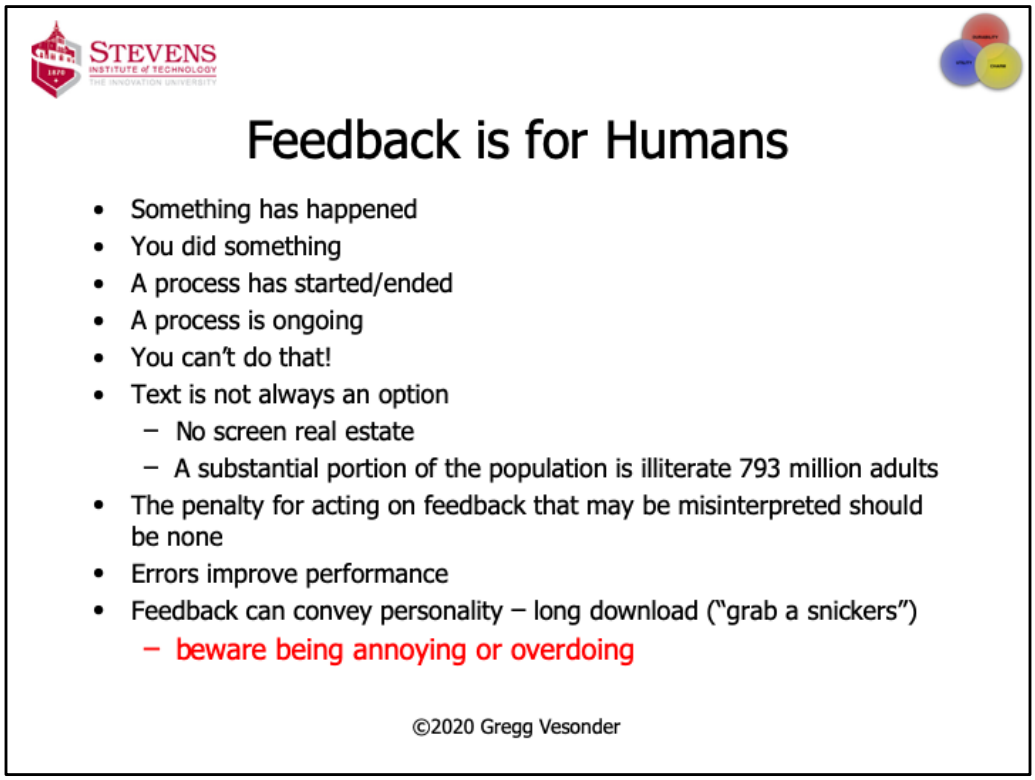
Feedback is necessary but you do not want to overburden the user with feedback – too much feedback overwhelms users and they often leave.



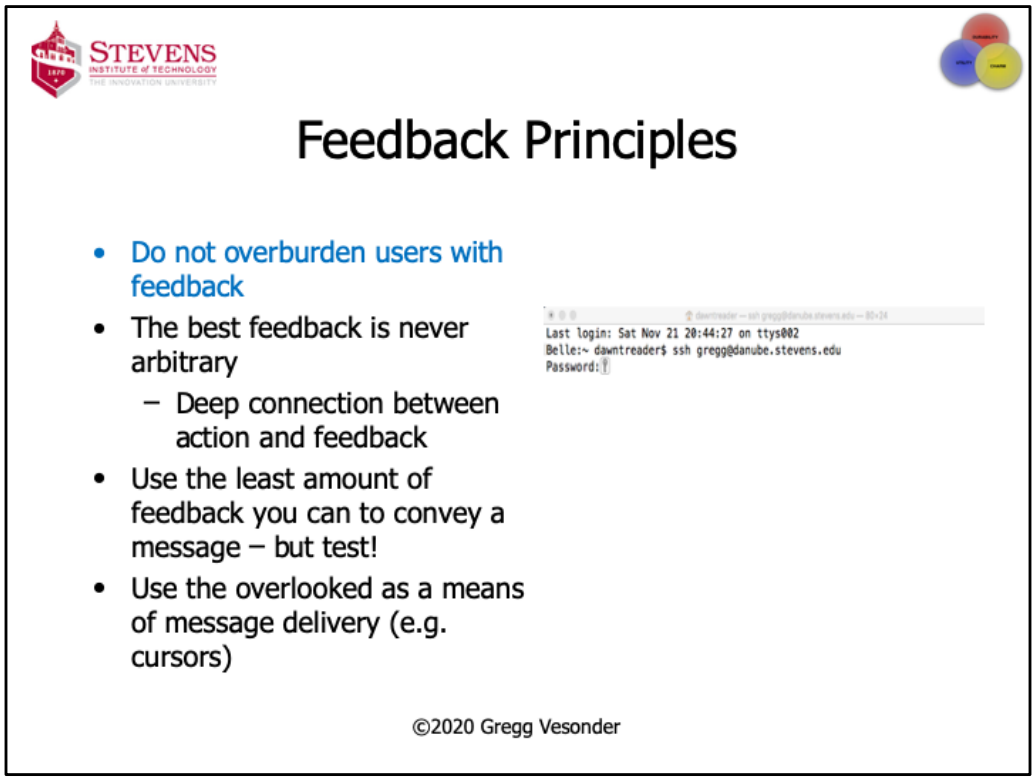
These are instances when feedback should occur. Any action that you do should be acknowledged by the system – not doing so will result in a user unsure of whether the system is still running. Similarly when something takes a long time the system should update the user on progress and insure that the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



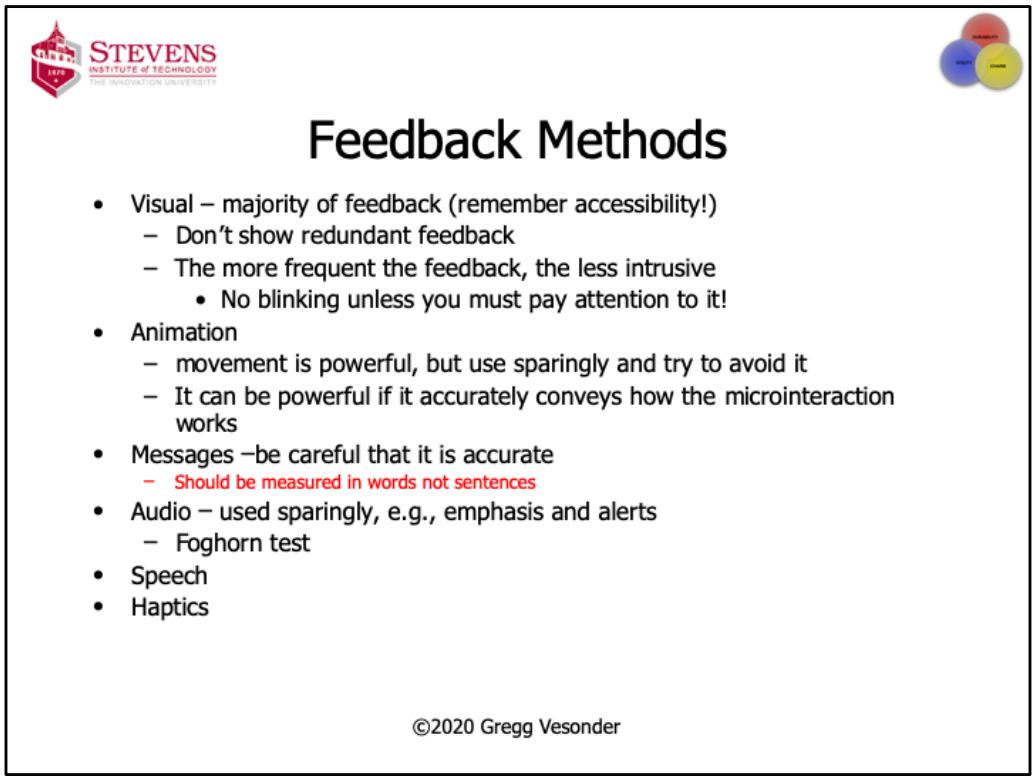
This is optional – feedback is necessary when you are in a moded text editor – one that has either an input or a command mode – vi or vim is a great example of this.



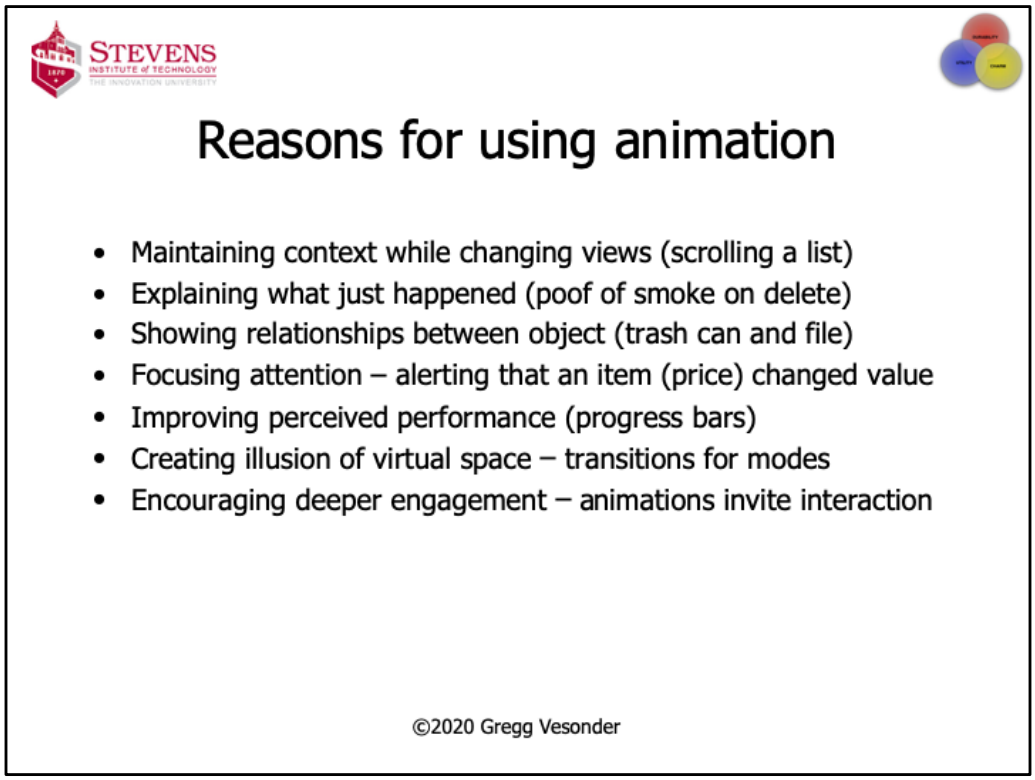
This reflects Norman's comment in the preface of the book. He stated that an error message is not that useful if it does not provide a remedy.. But there are challenges to feedback especially on the relatively small screens of smartphones and, if a basic services app (medical care, food, …) you must 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.



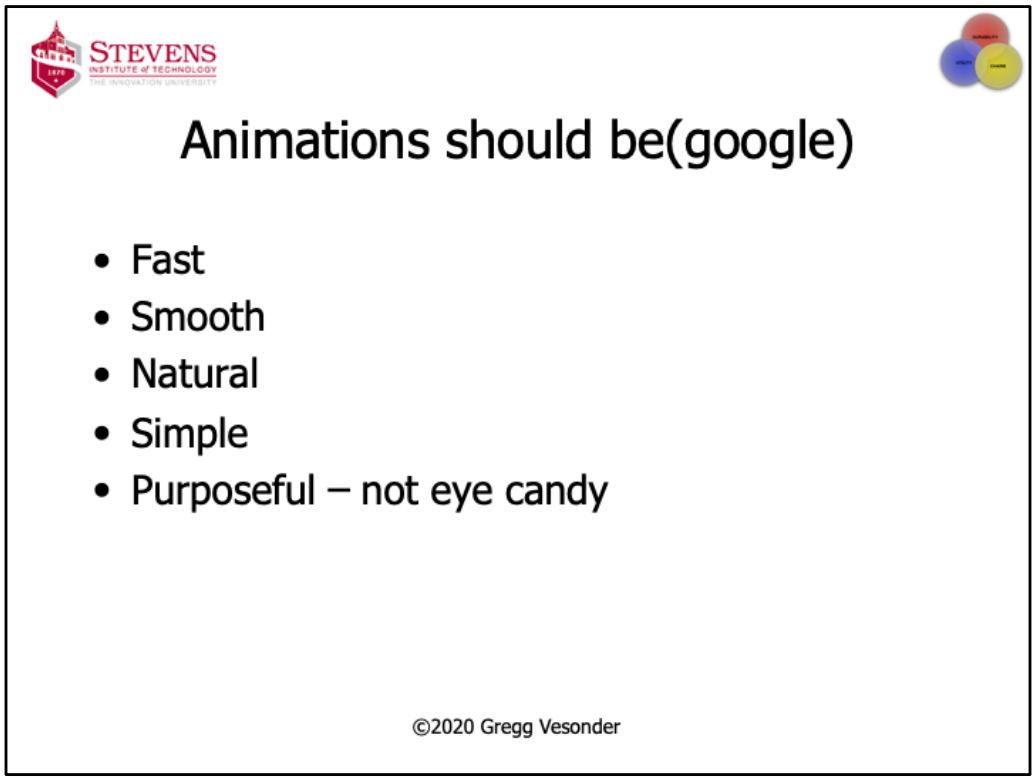
Use feedback judiciously – too much will be ignored and clutter your application. Using cursors for feedback is a great idea – for example when a password is asked on the command line screen of OS X (basically linux) the cursor changes to a key icon. And of course test the appropriateness and effectiveness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



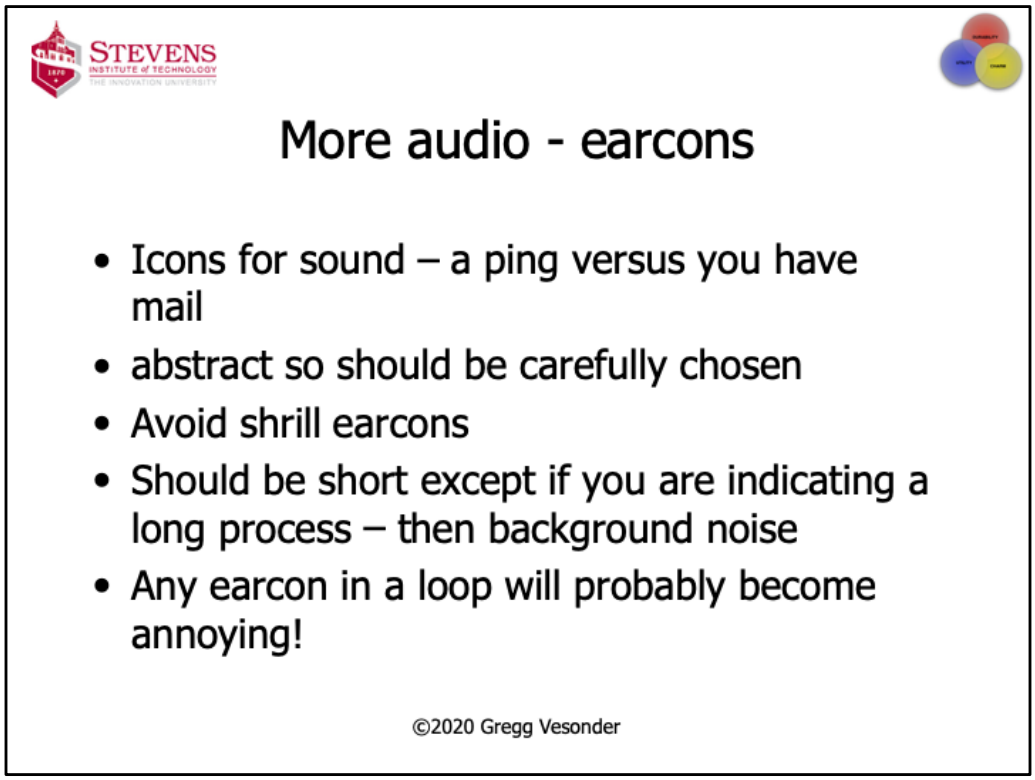
Of course feedback must take into account accessibility – almost all visual feedback should have a text version so that it can be translated to text to speech for visually impaired individuals. Feedback should be crisp measured in number of words rather than number of sentences. Audio is necessary for 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.



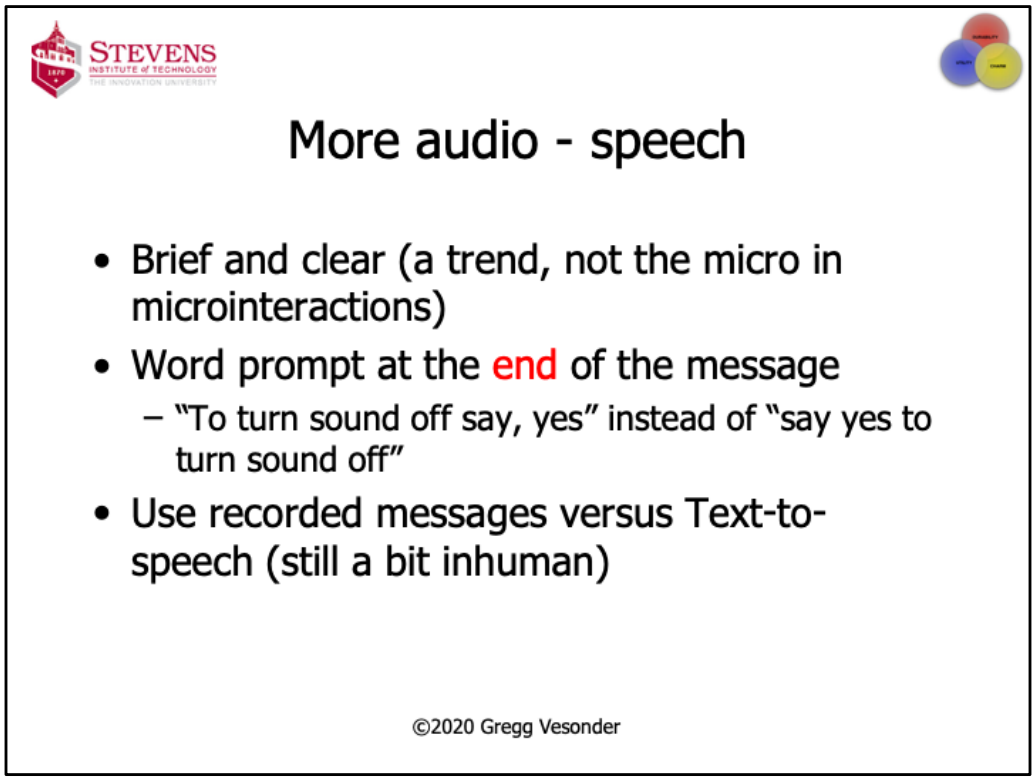
In the early days of the web animation was abused – try this website and be sure to enable sound : http://www.angelfire.com/super/badwebs/ . That web site should drive home many of the points made here and on the next page.



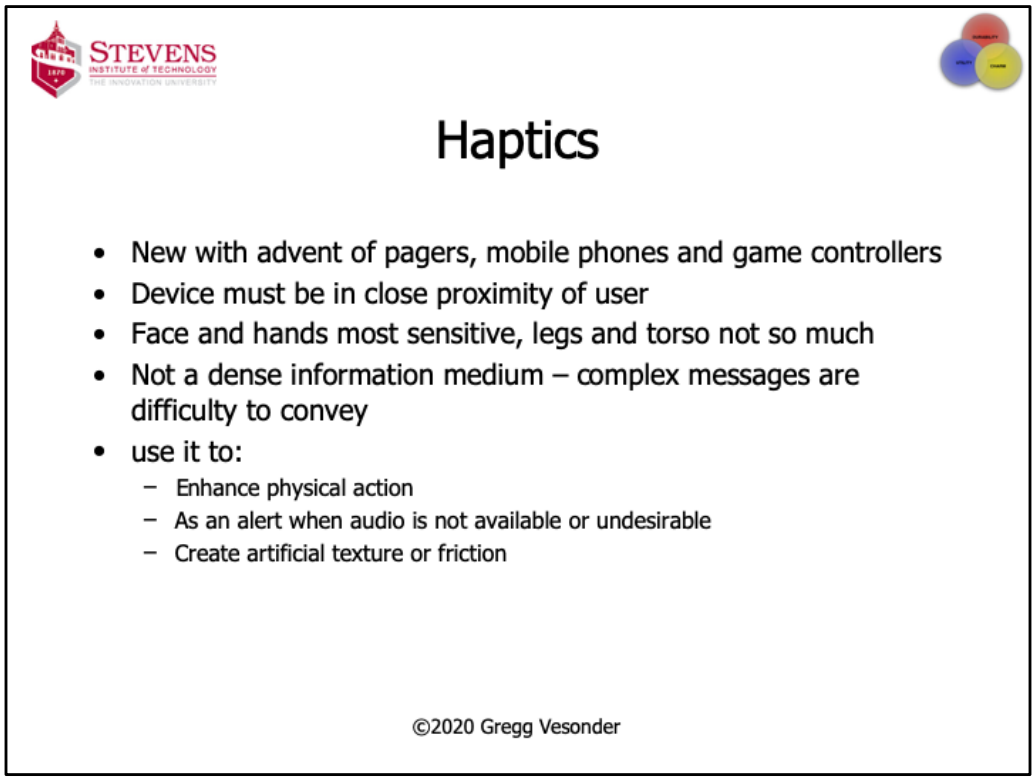
Google has provided a great set of advice for animations. My caution is make sure you really, really need the animation before doing it!



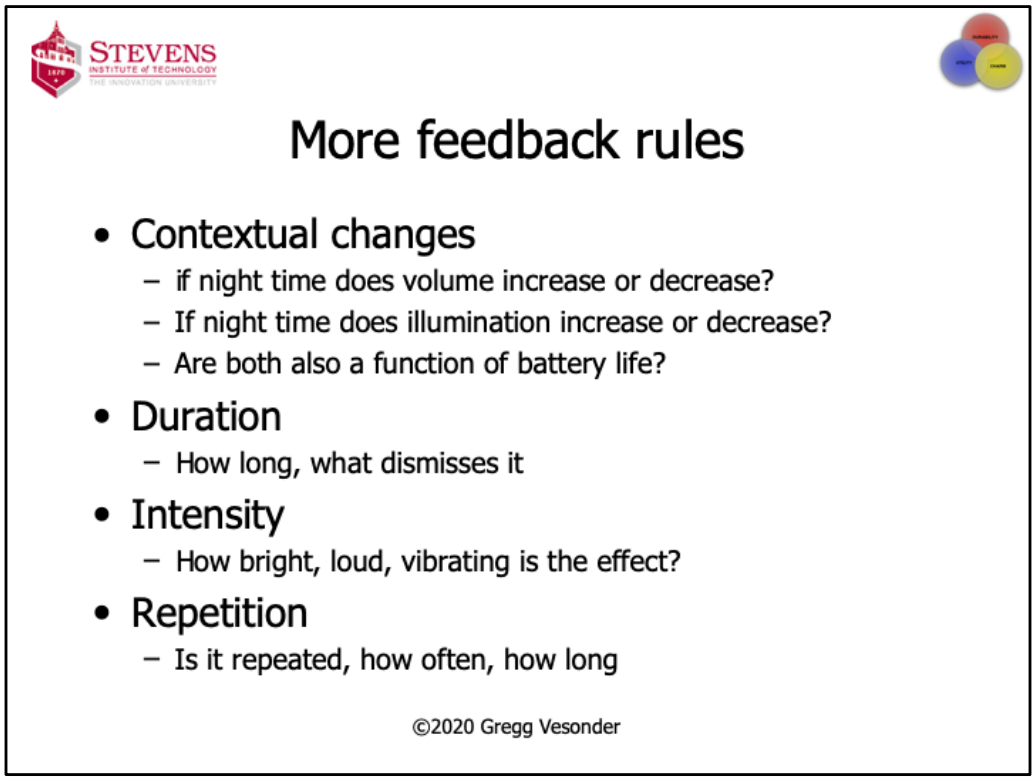
The website I provided a few slides back (slide 24) had obnoxious looping sound – please do not even think of dong that! Most systems and smart phones use a variety of earcons to indicate incoming mail, outgoing mail, new SMS messages, … . Often they are customizable and you have the option to disable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



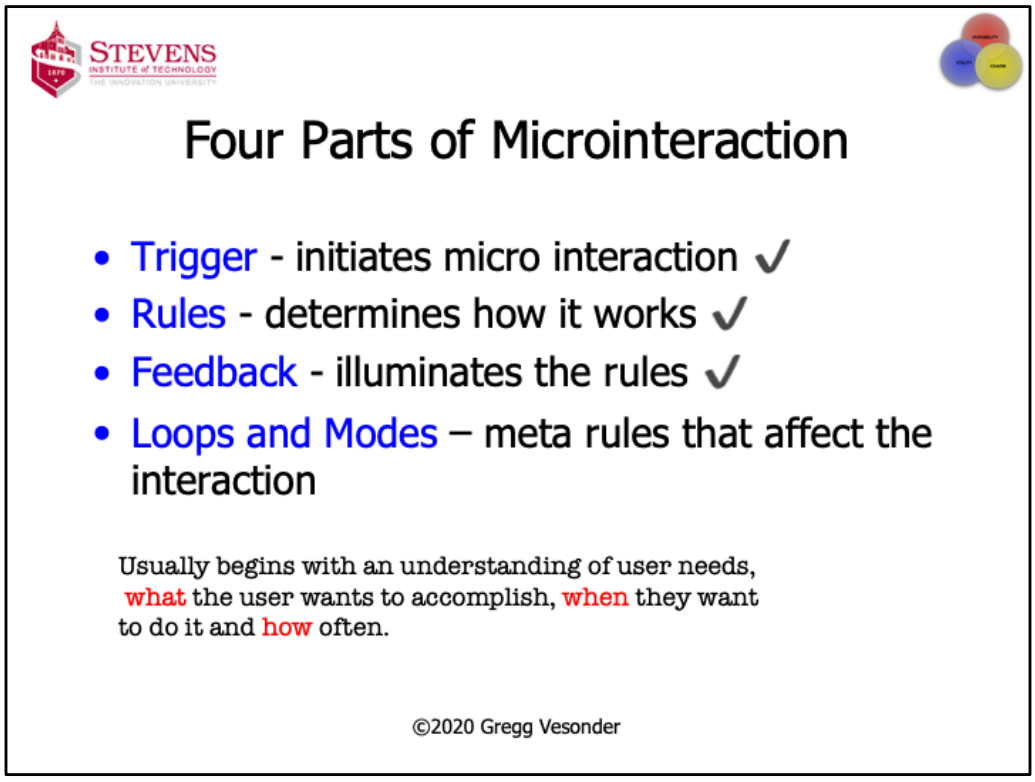
I am still not a fan of multiple sentences of generated audio. Note that, unlike pictures or text, speech is serial – you cannot hear the whole sentence said but only one word at a time. For that reason, even though it may result in inferior grammar, try to place the prompt choices at the end of the 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.



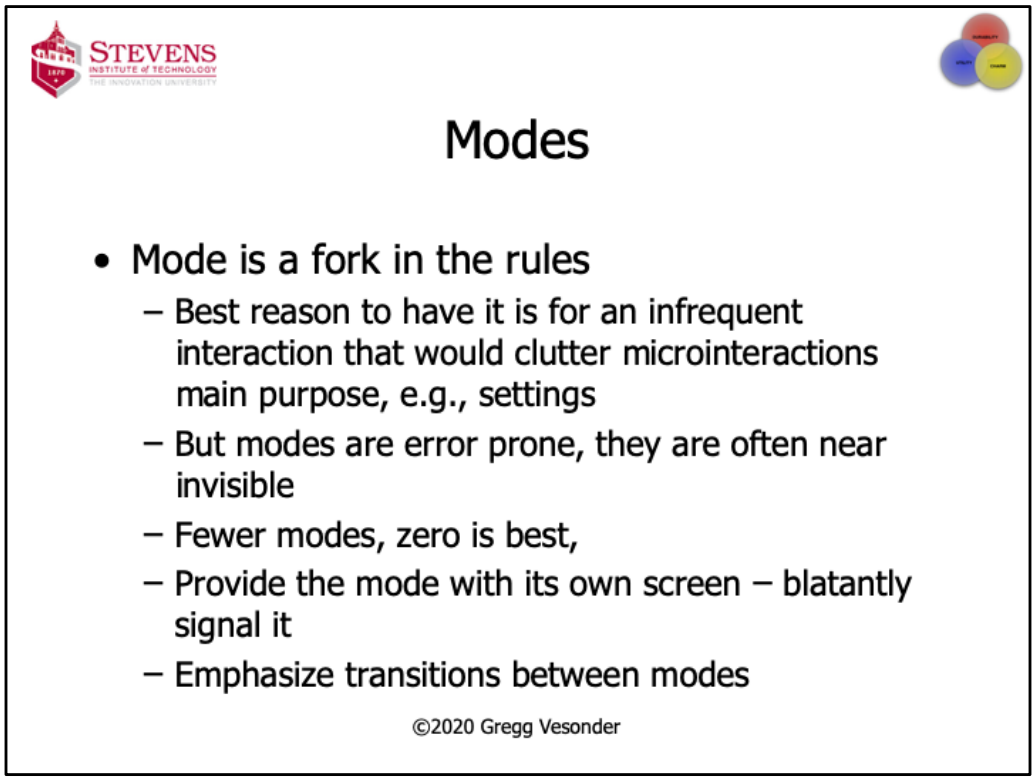
A major way my apple watch communicates with me is through haptics – it vibrates to get my attention – very handy. Since it is on my hand I am sensitive to the vibration, there are times that I do not even sense the vibration of phones in my pocket. Besides alerts it is especially effective in gaming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



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!

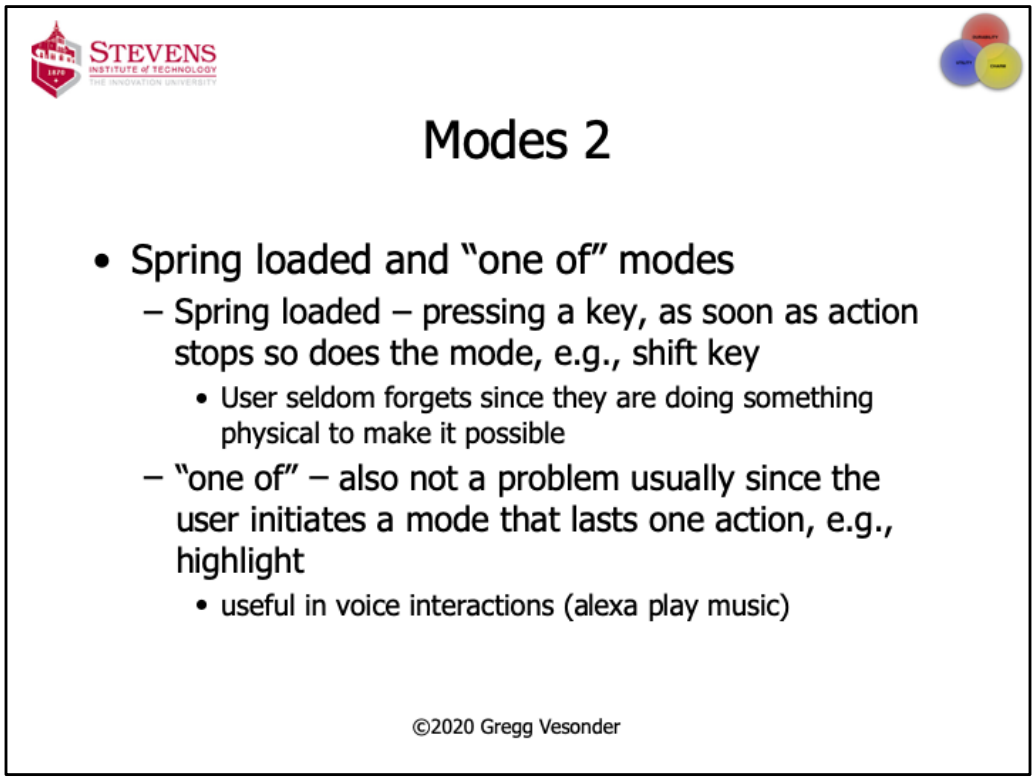


So we reviewed feedback and we are onto loops and modes that provide continuity to user interactions.

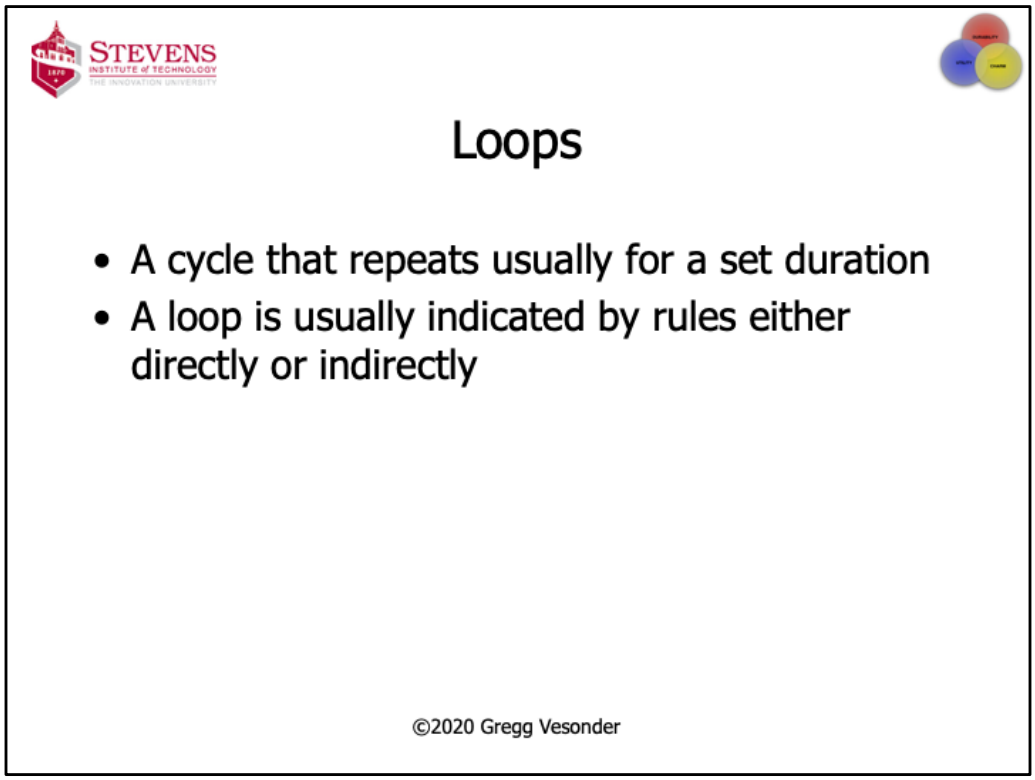


So a mode is a change of context. For example – a set of screens that provides settings for the application, e.g., volume, sound effects, control choice (keyboard arrows or mouse), … . Mode also can be used to differentiate between commands and text entry for example vim, a text editor. It is very important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

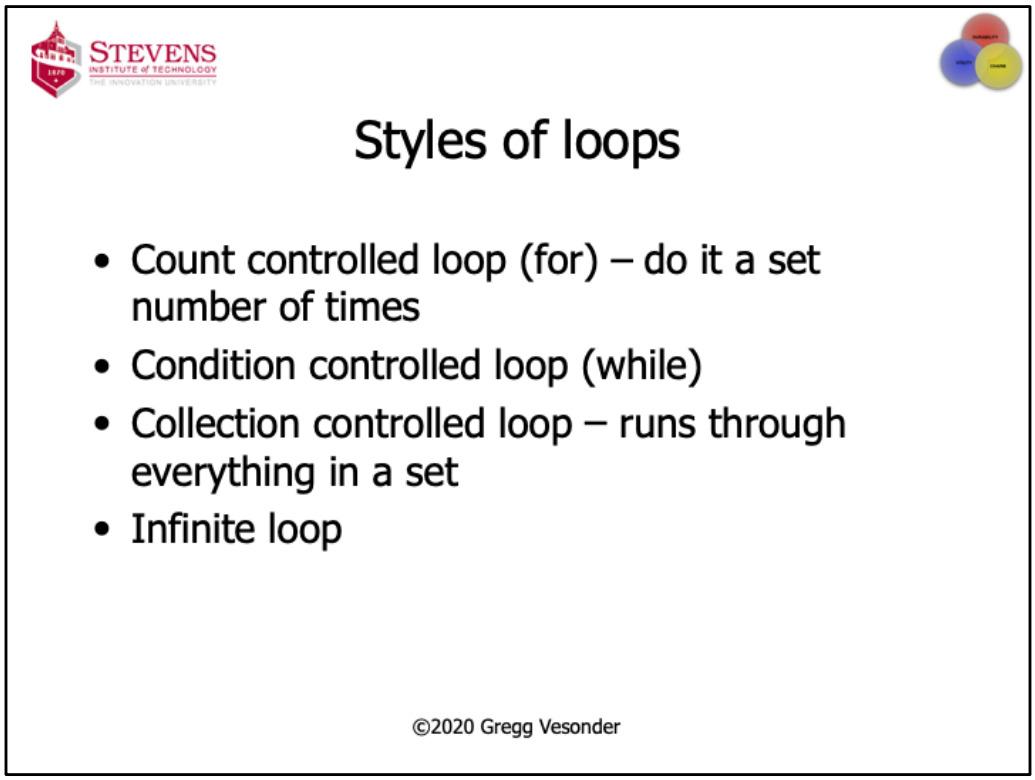
Although many folks encourage avoiding modes, they can be very helpful if done correctly.



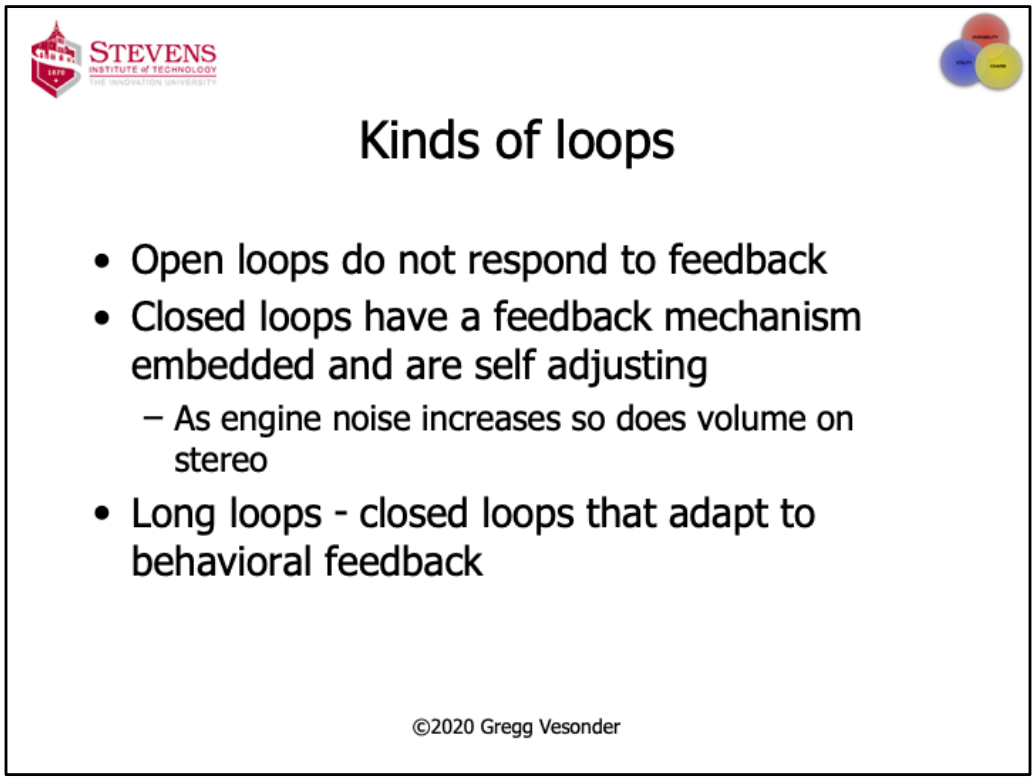
Note that often the shift key provides problems for passwords so even such a commonly used mode is error prone, especially if the spring loaded locks. Mode also is useful for specific actions and to protect privacy – for example using the amazon echo – the echo only listens when you use the word alexa.



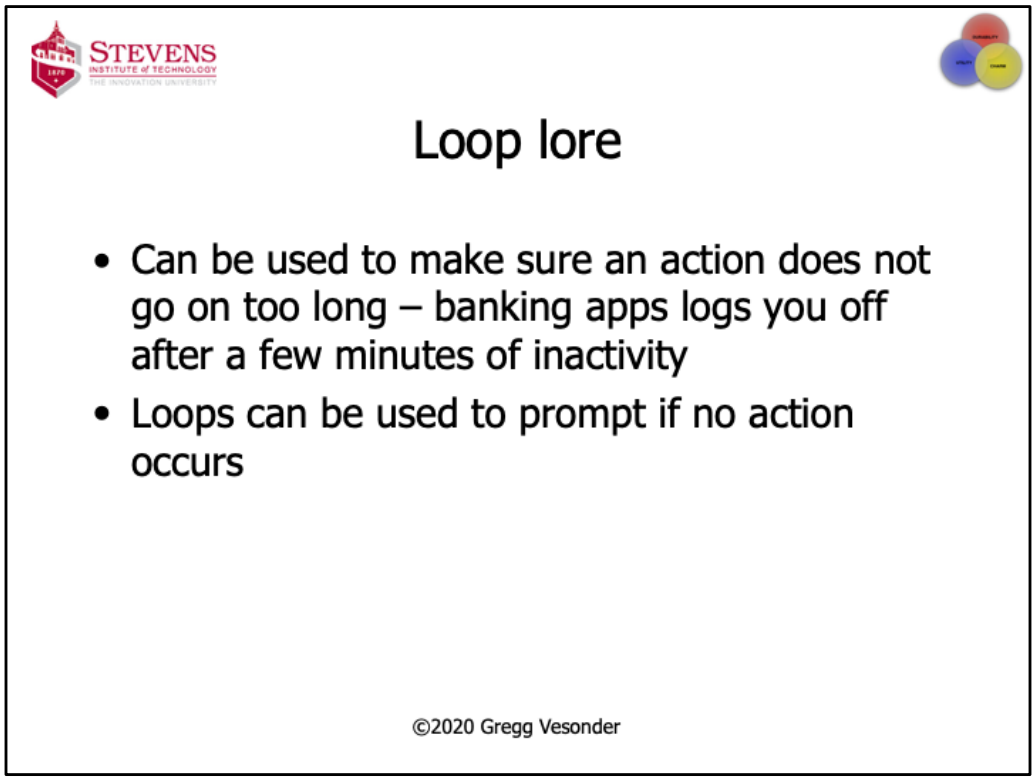
Loops are different it is an interval of time where actions are remembered or alerts set or … . Very analagous to typical program constructs such as *while* or *for* and may even use them in the implementation.



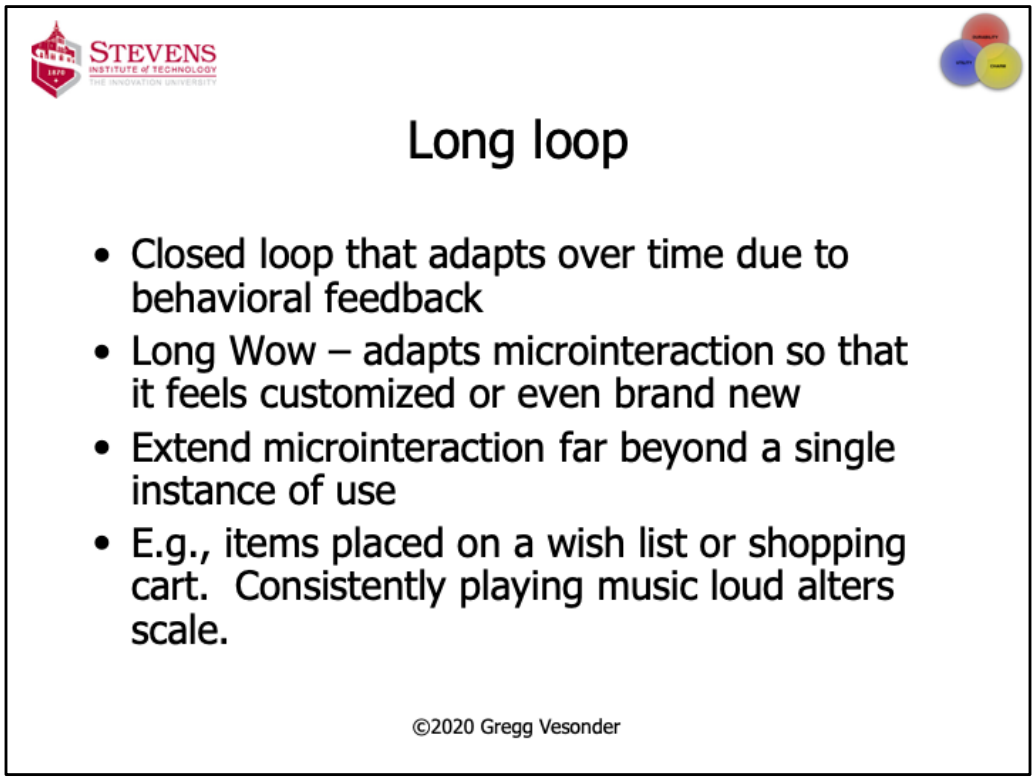
In particular an infinite loop while (true) do {} is great to begin understanding the user and adapting to the user behavior. You record what the user does and try to understand it perhaps readjusting menus in the applications once you understand what the user typically does.



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.



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.

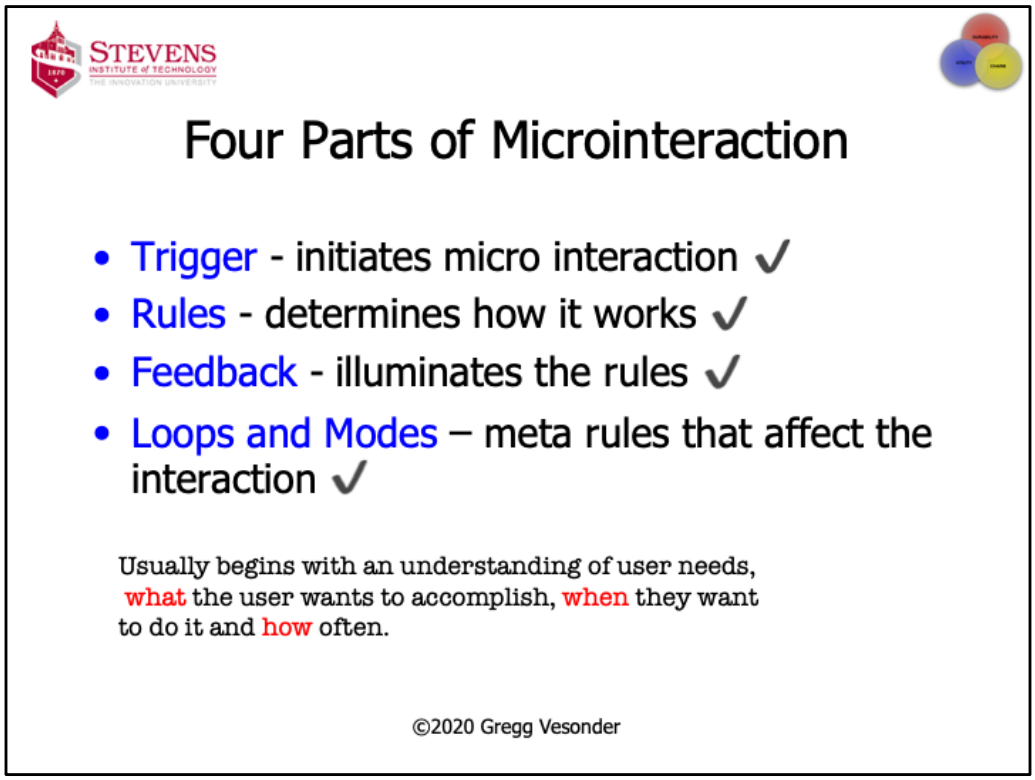


The long wow is being surprised occasionally when the long loop alters the application environment to be sensitive to user needs – for example reordering the menu reflecting how the
User is applying menu items. Other examples are reminding you about items placed in the shopping cart that you have not 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

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.



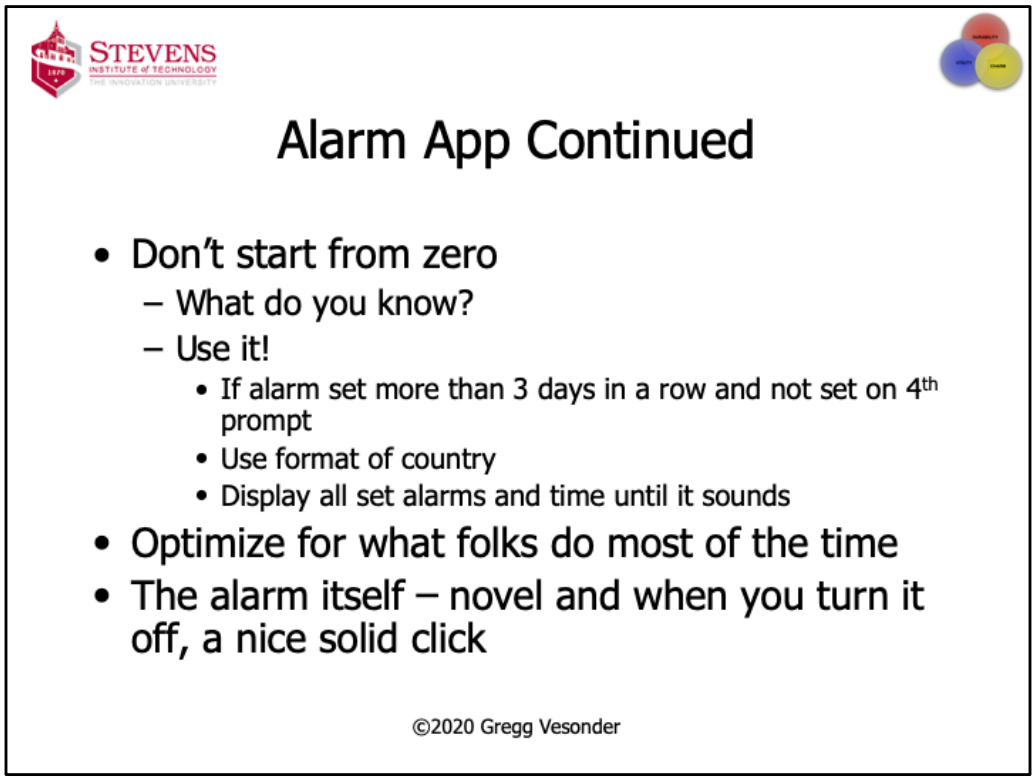
So we finished a whirlwind tour of the components of micro interaction. Let's explore some examples and how to design microinteractions more effectively.

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

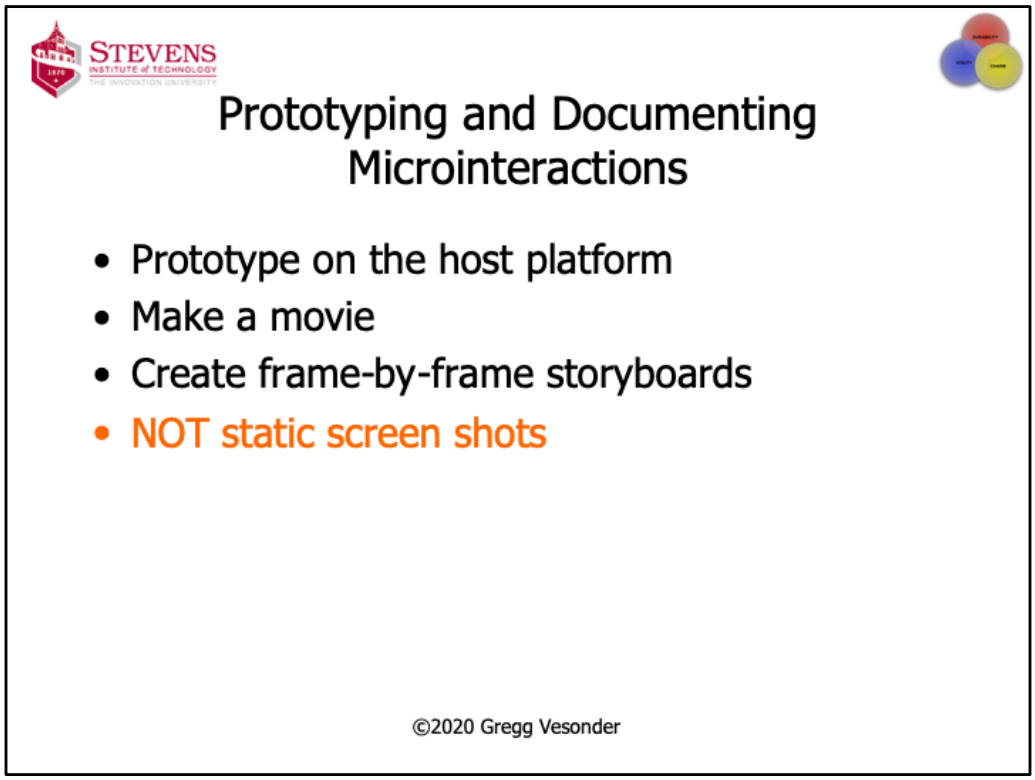
http://www.bigcheesebadges.com/number-2-stars-25mm-pin-button-badge-p-1600.html

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.

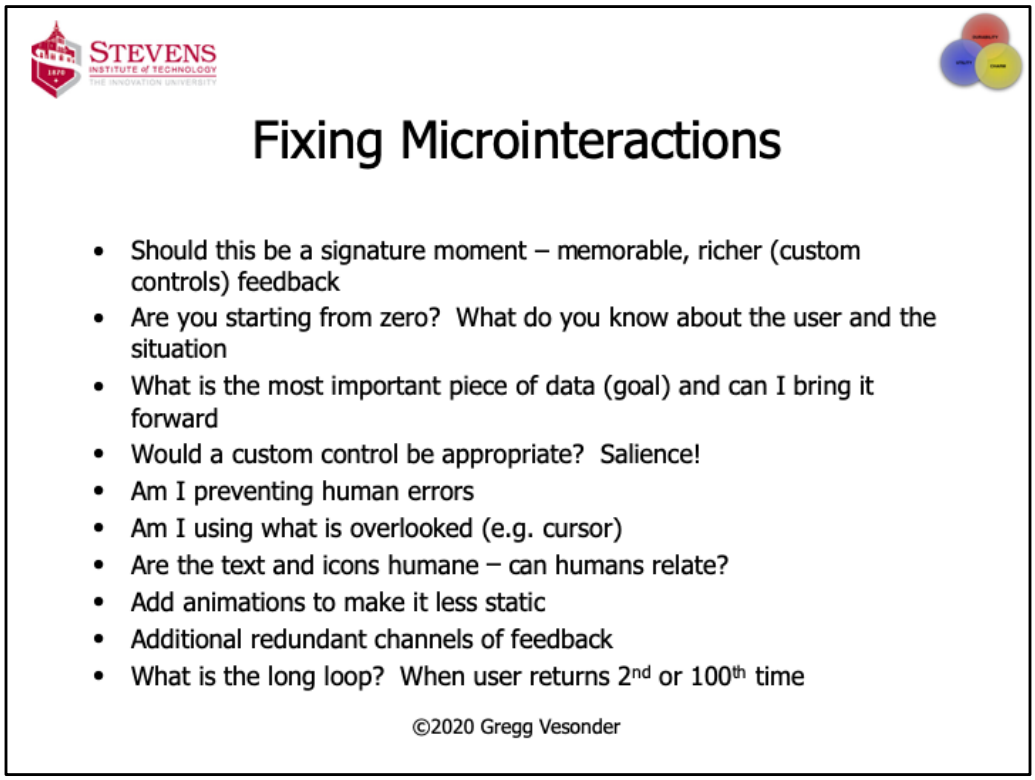


The "if alarm set more than 3 days and suddenly not set" is an example of another loop. Of course use the numeric format indigenous to the country. Buttons providing feedbacks – clicks are very satisfying just like a well made laptop clip.

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



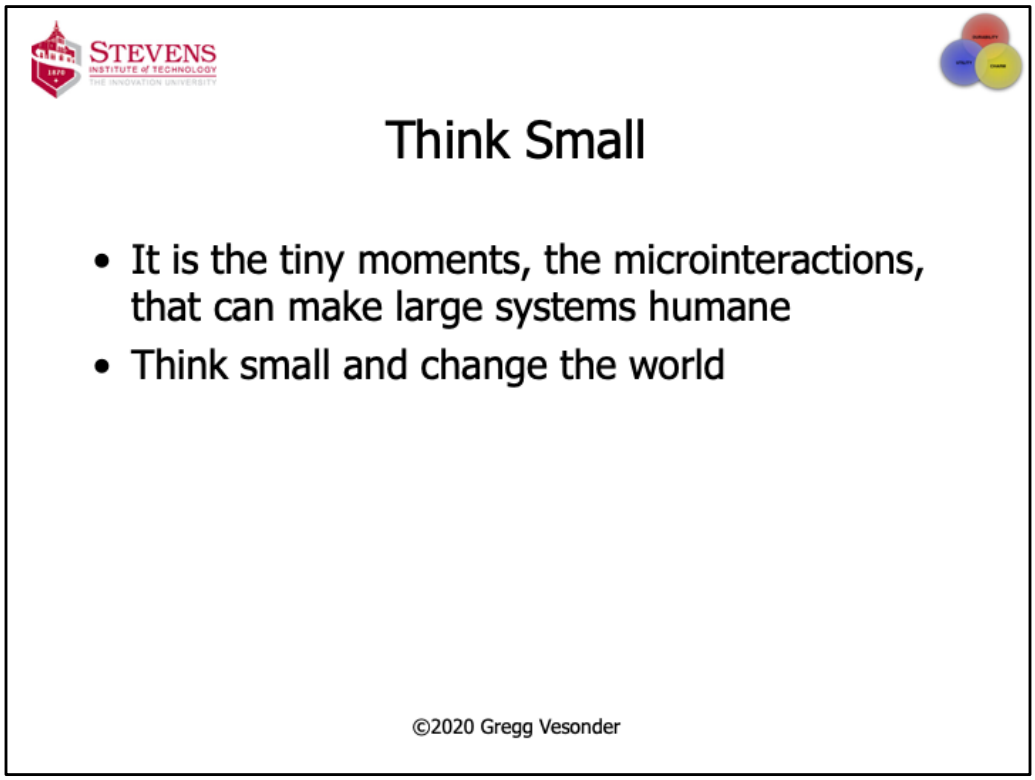
Microinteractions are by their very nature interactive and "alive." A static button does not make it – have it click and do something.



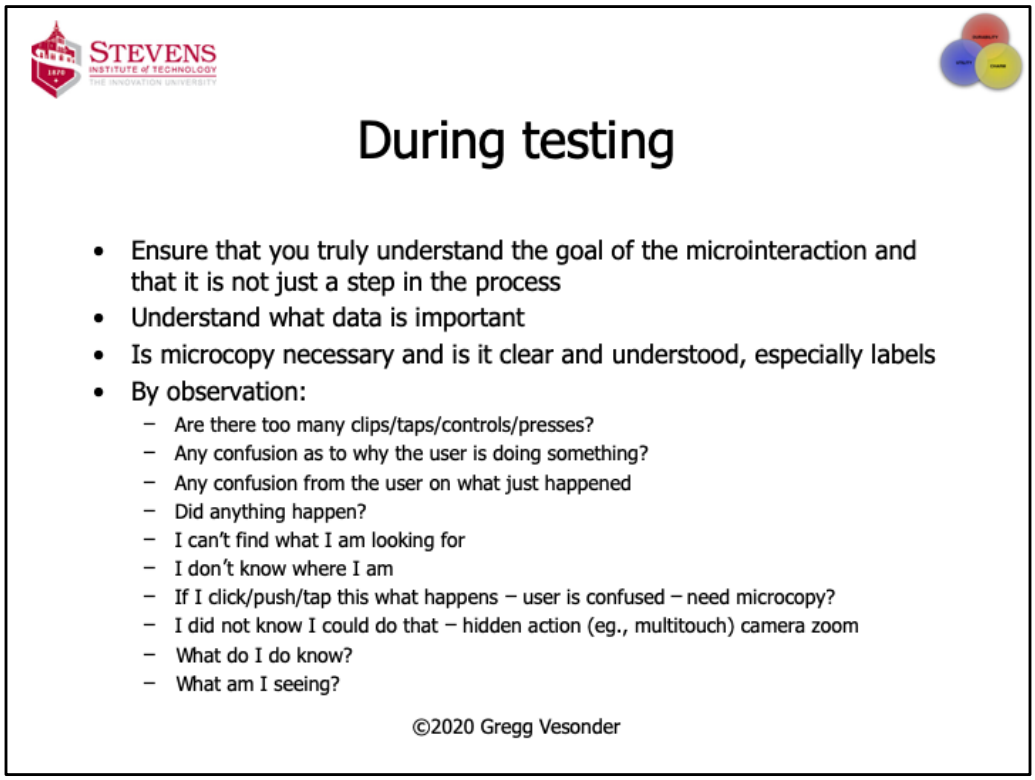
As is typical in many User Experience discussion there is always a set of suggestions, since there are few hard and fast rules.

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.

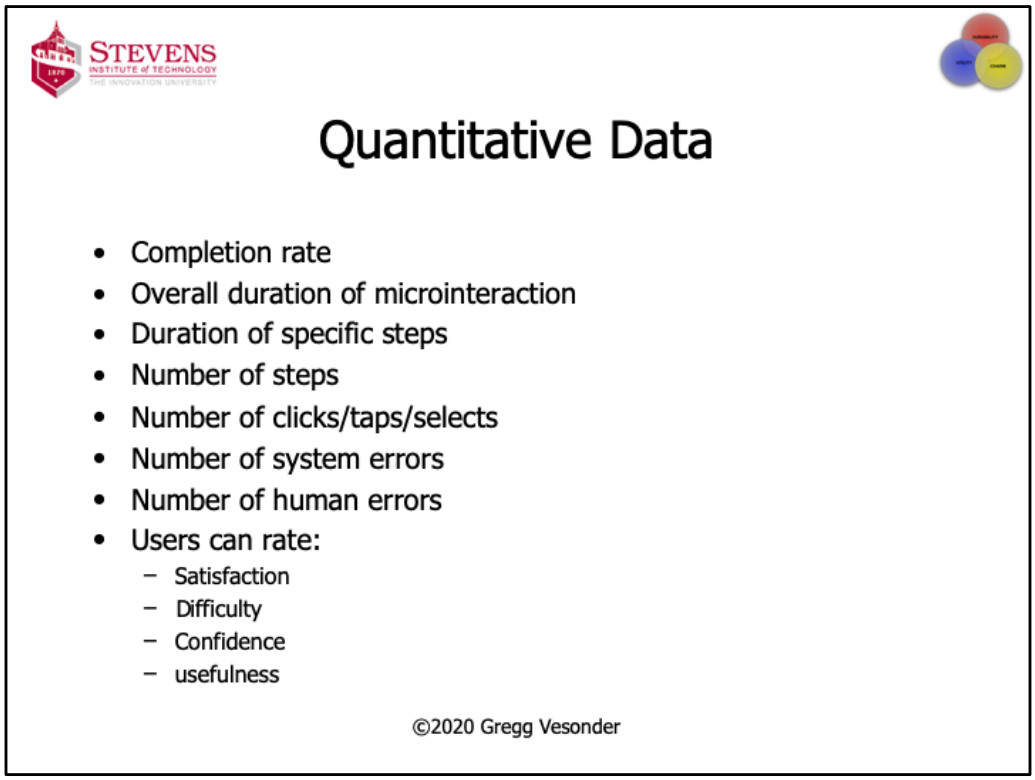


This summarizes the philosophy of the book, but wait, there is a bit more.



Again micrcopy 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.



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? I sit 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.