Review

• What are modes?
• Why do they cause user errors?
Quasimodes

- **Quasimode (QM):** temporary mode maintained kinesthetically
  - Shift for uppercase (vs. Caps Lock), Mac menus
  - no mode errors, but don’t overuse them (max. 4–7 QMs)
  - bad: Control↓ Alt↓ Shift↓ Esc↓ q↑ ↑ ↑ ↑

- **Habituating feature:** successfully operated by the blind
  - e.g., Canon Cat paragraph styles

- **Adaptive menu / palette**
  - place most (recently) used item at top of menu
  - 2 approaches: remove chosen item from list or duplicate it
  - e.g., Vellum (CAD program): adaptive palette and item at accustomed place

- 2 fundamental kinds of input to computer
  - *create content* or *control system*
  - rule of thumb: use quasimodes for control, but not for content
  - Why not the other way around?
Noun-Verb vs. Verb-Noun style

• Many commands apply an action to an object
  – e.g., change font of paragraph
• Noun-verb recommended by interface guidelines
  – command is locus of attention (no modes)
  – only one change of locus of attention
  – no cancel feature needed
• Verb-noun used for palettes
  – e.g., brush styles in paint programs
  – frequent mode errors but appearance is locus of attention
  – pure noun-verb model possible yet unnatural
• Case study: order form
Visibility and Affordances

- **Visible** interface feature
  - accessible to a human sense organ or in short-term memory

- **Invisible** feature
  - need to memorize that feature exists, use help system… (mapping problem)

- **Affordance** (DIS I, Norman)
  - “…indicate what parts to operate and how, how the user is to interact with the device…”
  - e.g., volume turning knob, push buttons, slots, balls
  - depends on user’s experience and background, culture, context

- Optimize cognitive properties of interface
  - make all functions and the methods of operating them apparent by looking
  - visible feature should provide a recognizable affordance (icons?)

- Example: BART ticket system
Monotony

• Many methods execute the same command in today’s UIs
  – e.g., menus & short-cuts, cut & paste vs. select & drag, autopilots
  – *backward* compatibility, appropriateness, managerial indecision
  – leads to complex products, errors, increased training time, costs…

• **Modeless** interface
  – gesture \( g \) always results in action \( a \), but gesture \( h \) may too

• **Monotonous** interface
  – only one gesture for one particular command
  – monotony often happens spontaneously

• Modeless and monotonous interface
  – one-to-one correspondence between cause and effect
  – interface fades from user’s consciousness (keep attention at task)
  – habitual and addictive product use
Beginner or Expert?

• “We’re humans first, beginners or experts second.”
  – consider cognitive capabilities and demands of task
• Common myth: different interfaces for user’s experience
  – only true for some systems, too many features to know
• Adapting to user’s level of expertise usually fails
  – How should system know your rate of learning or memory decay?
  – e.g., Windows 2000 changes order / set of menu items
• View interface through eyes of an individual
  – brief period of (conscious) learning: simplicity and visibility
  – long period of (automatic) routine: efficiency is important
• Satisfy all users’ needs with one mechanism
Quantitative analyses of interface designs

- **GOMS** (Card, Moran, and Newell 1983)
  - model of goals, operators, methods, selection rules
  - predict time an experienced worker needs to perform a task in a given interface design
- **Keystroke-level GOMS model** (simplified version)
  - comparative analyses of tasks that use GID and keyboard
  - correct ranking of performance times using different interface designs
- **CPM-GOMS** (critical path method)
  - computes accurate absolute times
  - considers overlapping time dependencies
- **NGOMSL** (natural GOMS language)
  - considers non-expert behavior (e.g., learning times)
Keystroke-level model

- Execution time for a task = sum of times required to perform the serial elementary gestures of the task
- Typical gesture timings
  - **Keying** $K = 0.2$ sec (tap key on keyboard, includes immediate corrections)
  - **Pointing** $P = 1.1$ sec (point to a position on display)
  - **Homing** $H = 0.4$ sec (move hand from keyboard to GID or v.v.)
  - **Mentally preparing** $M = 1.35$ sec (prepare for next step, routine thinking)
  - **Responding** $R$ (time a user waits for the system to respond to input)
- Responding time $R$ effects user actions
  - causality breakdown after 250 ms
  - give feedback that input received & recognized
Keystroke-level calculation

1. List required gestures
   – e.g., HK = move hand from GID to keyboard and type a letter

2. Compute mental preparation times Ms
   – difficult: user stops to perform unconscious mental operations
   – placing of Ms described by rules

3. Add gesture timings
   – e.g., HMPK = H + M + P + K = 0.4 + 1.35 + 1.1 + 0.2 = 3.05 sec

• Rule terminology
  – string: sequence of characters
  – delimiter: character marking beginning (end) of meaningful unit
  – operators: K, P, and H
  – argument: information supplied to a command
Rules for placing Ms

- Rule 0, initial insertion for candidate Ms
  - insert Ms in front of all Ks
  - place Ms in front of Ps that select commands, but not Ps that select arguments for the commands

- Rule 1, deletion of anticipated Ms
  - delete M between two operators if the second operator is fully anticipated in the previous one (e.g., PMK → PK)

- Rule 2, deletion of Ms within cognitive units
  - in a string of MKs that form a cognitive unit, delete all Ms except the first (e.g., “Helen of Troy”, 745.8)
Rules for placing Ms

• Rule 3, deletion of Ms before consecutive terminators
  – if K is redundant delimiter at end of a cognitive unit, delete the M in front of it, e.g., )’

• Rule 4, deletion of Ms that are terminators of commands
  – if K is a delimiter that follows a constant string then delete the M in front of it (not for arguments or varying strings)

• Rule 5, deletion of overlapped Ms
  – do not count any M that overlaps an R (e.g., user waiting for computer response)
Exercise: temperature converter

• Convert from degrees Fahrenheit (F) to Celsius (C) or v.v., requests equally distributed
• Use keyboard or GID to enter temperature
• Assume active window awaiting input, an average of four typed characters (including point and sign), and no typing errors

• Task: create and analyze your own interface!
The dialog box solution with radio buttons…
...and its keystroke-level model

- **Case 1: select conversion direction**
  - move hand to GID, point to desired button, click on radio button (HPK)
  - move hands back to keyboard, type four characters, tap enter (HPKHKKKK)
  - Rule 0 (HMPMKHKMKMKMKMKMK)
  - Rule 1, 2, 4 (HMPKHKKKKKMK)
  - Estimated time = 7.15 sec

- **Case 2: correct conversion direction already selected**
  - MKKKKKMK = 3.7 sec

- **Average time** $= (7.15 + 3.7) / 2 = 5.4$ sec
Measuring interface efficiency

• How fast can you expect an interface to be?
• *Information* as quantification of amount of data conveyed by a communication (Information theory)
  – e.g., speech, messages sent upon click…
• Lower bound on amount of information required for task is independent of interface design
• **Information (theoretic) efficiency** $E = \min. \text{amount of information required for task} / \text{amount of information supplied by user}$
  – $E = 0..1$ (e.g., $E = 0$ for providing unnecessary information)
• **Character efficiency** $= \min. \text{number of characters required for task} / \text{number of characters entered in interface}$
Quantify amount of data

- Information is measured in bits
  - 1 bit represents choice between 2 alternatives
- n equally likely alternatives
  - total information amount: $\log_2(n)$
  - information per alternative: $(1/n)\log_2(n)$
- n alternatives with different probabilities $p(i)$
  - information per alternative: $p(i)\log_2(1/p(i))$
  - total amount = sum over all alternatives
- Consider situation as a whole
  - probability of messages required
  - information measures freedom of choice (information ≠ meaning)
Example: temp. converter

- Average of 4 typed float chars, 25% negative values
  - -.dd and -d.d (each 12.5% and 100 values)
  - .ddd, d.dd, dd.d (each 25% and 1000 values)
  \[ \approx 11.4 \text{ bits/message, simple approach: } 4 \log_2(12) \approx 14 \text{ bits} \]

- Information efficiency
  - 128 keys standard keyboard (5 bits/key): \( E \approx 55\% \ (11/20) \)
  - 16 keys numeric keypad: \( E \approx 60\% \)
  - 12 keys dedicated keypad: \( E \approx 70\% \)

- Keystroke efficiency
  - type C or F, value, enter: M K K K K M K \( \Rightarrow \) 3.9 sec (char. eff. 67 \%)
  - type value, then C or F: M K K K K M K \( \Rightarrow \) 3.7 sec (char. eff. 80\%)
  - bifurcated: M K K K K = 2.15 sec (char. eff. 100 \%)
Fitts’ law

- Time to acquire a target is a function of distance (D) and size (S)
  - one dimensional, rapid movement

- Movement time $MT = a + b \log_2(D/S + 1)$ in ms
  - empirical constants $a$, $b$ (e.g., $a=50$, $b=150$)
  - $\log_2(D/S + 1)$ measures difficulty of task in bits
Hick’s law

- Predict choosing time of one among $n$ alternative actions
- Actions equally distributed
  - $T = a + b \log_2(n + 1)$ in ms
- Actions with different probabilities
  - $T = a + b \sum p(i) \log_2(1/p(i) + 1)$
- Coefficients $a$ and $b$ as for Fitts’ law
  - influenced by habituation and other factors
- Example: choosing from a menu with 8 items is faster than choosing from two menus with 4 items each
  - $a + b \log_2(9) < 2(a + b \log_2(5))$