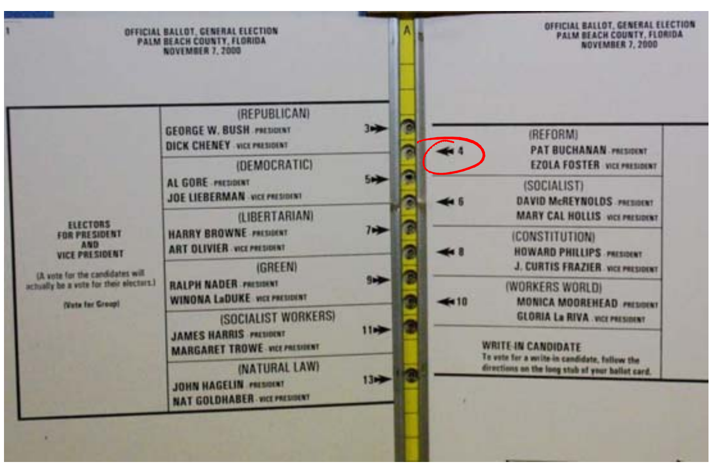
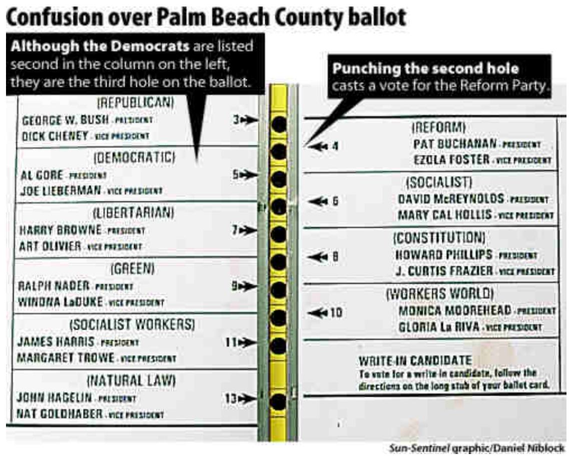
Human Information Processing (KLM, GOMS, Fitts', Hick's)

CS160: User Interfaces

Maneesh Agrawala





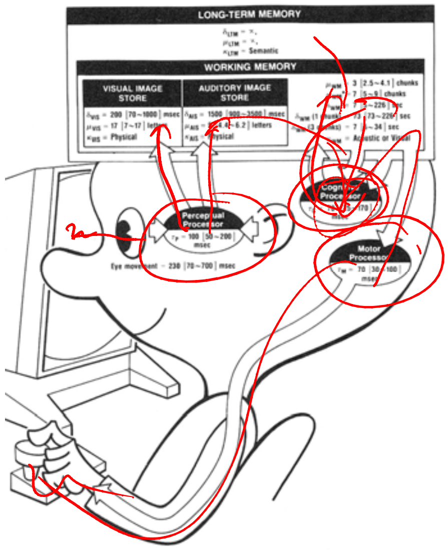
- Poor layout – easy to vote for wrong person
- Punch through design leads to hanging chads
- More analysis by Bruce Tognazzini: http://www.asktog.com/columns/042ButterflyBallot.html

Assignment (due Mar 11)

Low-Fidelity Prototype

- Identify project mission statement
- Create low-fidelity prototype that supports 3 tasks
  - 1 easy, 1 moderate, 1 difficult task as found in the last assignment
- Test the prototype with target users
  - No one from this class
  - Not your friends

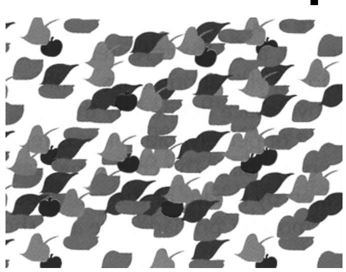
Review: Human Info Processor

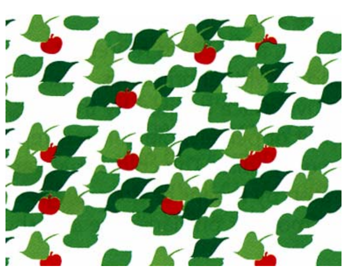


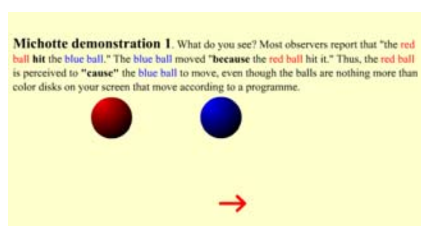
5 Parts
- Perceptual
- Cognitive
- Motor (will discuss today)
- Working memory
- Long-term memory

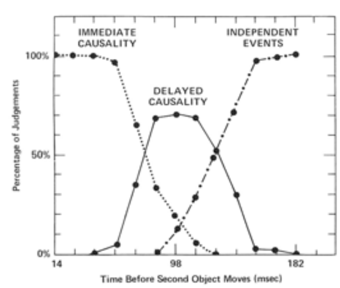
Unified model
- Probably inaccurate
- Predicts perf. well
- Very influential

Review: Pop-Out and Causality

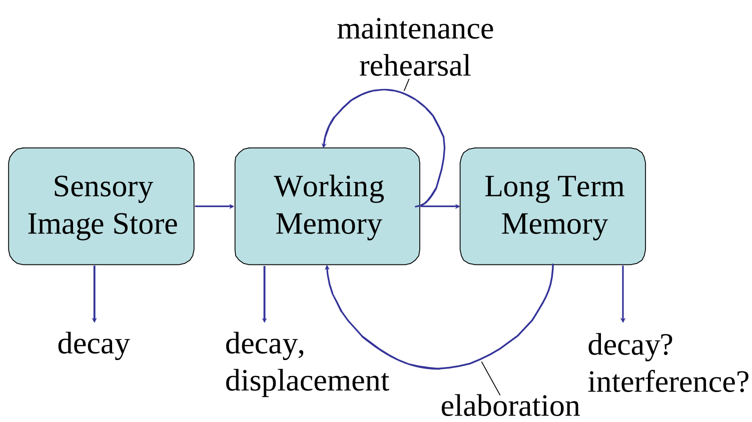








Stage Theory



Recognition over Recall

Recall
- Information reproduced from memory

Recognition
- Presentation of info helps retrieve info (helps remember it was seen before)
- Easier because of cues to retrieval

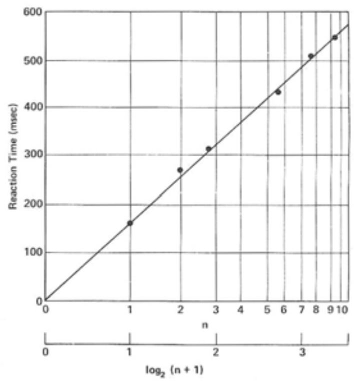
Topics

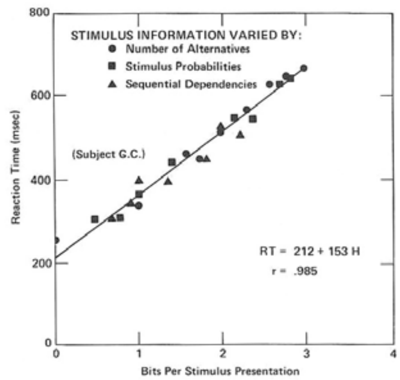
- Decision Making and Learning
- Fitts' Law
- GOMS and KLM

Decision Making and Learning

Hick's Law

Cost of taking a decision: $T = a + b\log_2(n+1)$





Power Law of Practice

- Task time on the nth trial follows a power law

$$T_n = T_1 n^{-a} + c$$

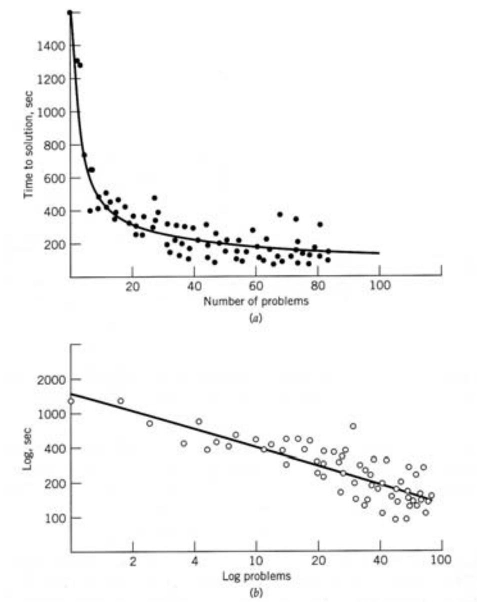
where a = .4, c = limiting constant

  - You get faster the more times you do it!

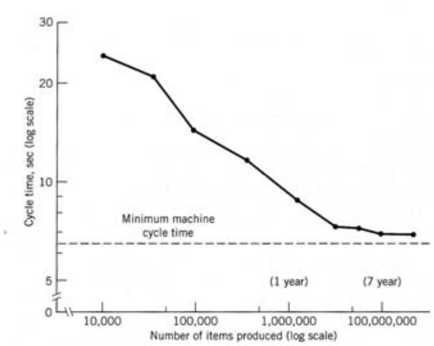
Applies to skilled behavior (sensory & motor)

Does not apply to

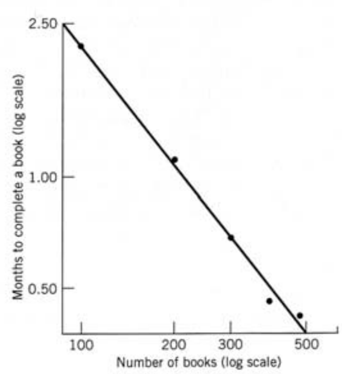
- Knowledge acquisition
- Improving quality



Problem solving



Manual skills

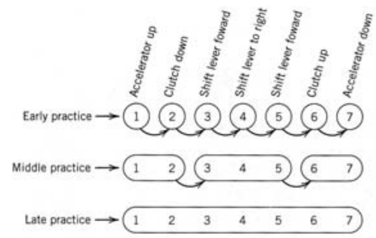


Writing books

Stages of skill acquisition

Example: Using a manual transmission

- Cognitive
  - Verbal representation of knowledge
- Associative
  - Proceduralization
    - Form of chunking
- Autonomous
  - More and more automated
  - Faster and faster
  - No cognitive involvement
    - Difficult to describe what to do



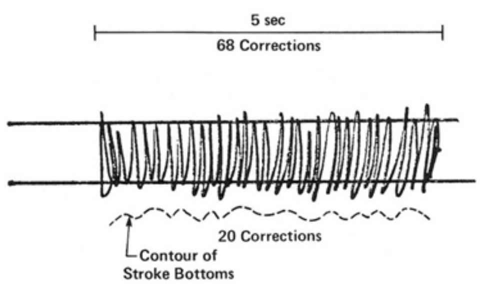
Fitts' Law

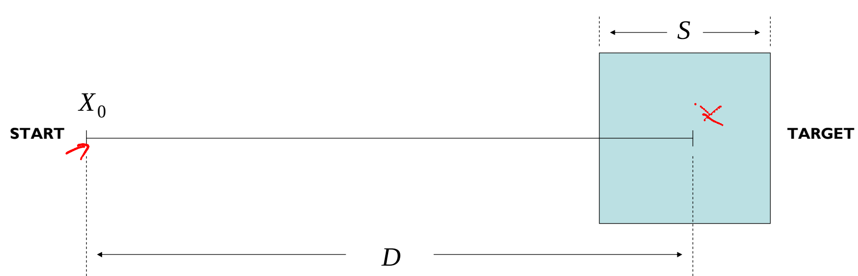
Motor Processor

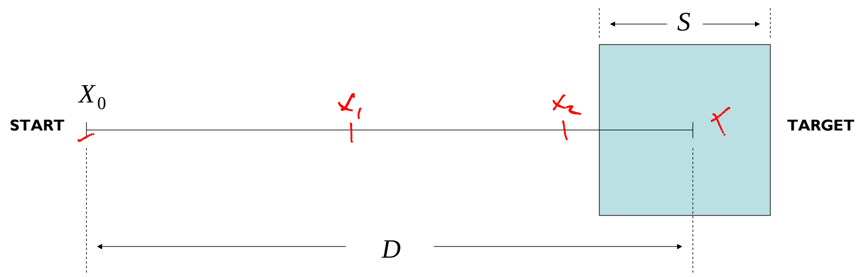
Receive input from the cognitive processor

Execute motor programs

- Pianist: up to 16 finger movements per second
- Point of no-return for muscle action







Hand movement based on series of microcorrections

X_i = remaining distance after ith move

relative movement accuracy remains constant → $\frac{X_i}{X_{i-1}} = \varepsilon$

$\frac{x_1}{x_0} = \varepsilon$

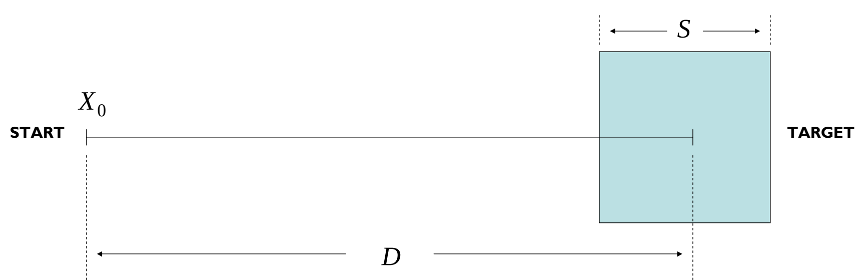
$\frac{x_2}{x_1} = \varepsilon < 1$

$x_1 = \varepsilon x_0 = \varepsilon D$

$x_2 = \varepsilon(\varepsilon x_0) = \varepsilon^2 D$

$x_n = \varepsilon^n D \leq \frac{S}{2}$

$n = \log_\varepsilon\left(\frac{S}{2D}\right) = I_m \log_2\left(\frac{2D}{S}\right)$



Fitts' Law

$$T = a + b\log_2(D/S + 1)$$

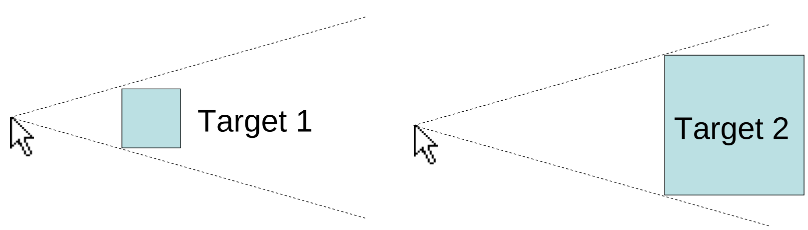
a, b = constants (empirically derived)
D = distance
S = size

ID is Index of Difficulty = $\log_2(D/S+1)$

- Models well-rehearsed selection task
- T increases as the ***distance*** to the target increases
- T decreases as the ***size*** of the target increases

Considers Distance and Target Size

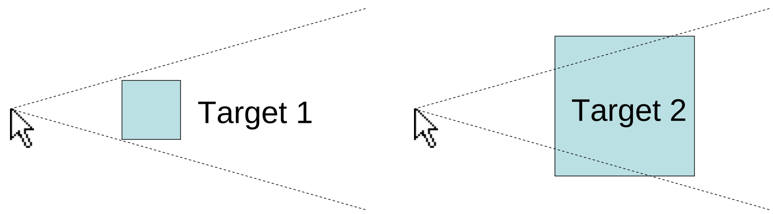
$$T = a + b\log_2(D/S + 1)$$



Same ID → Same Difficulty

Considers Distance and Target Size

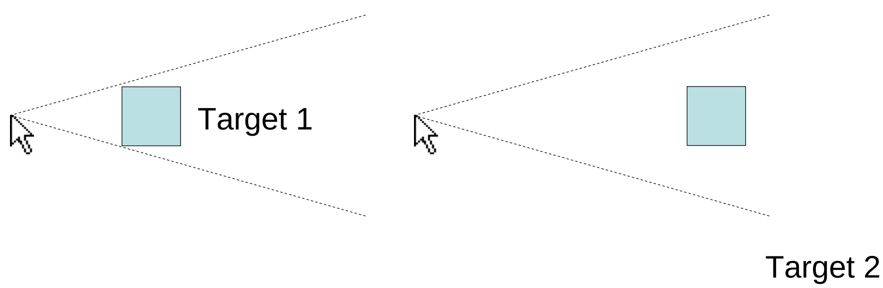
$$T = a + b\log_2(D/S + 1)$$



Smaller ID → Easier

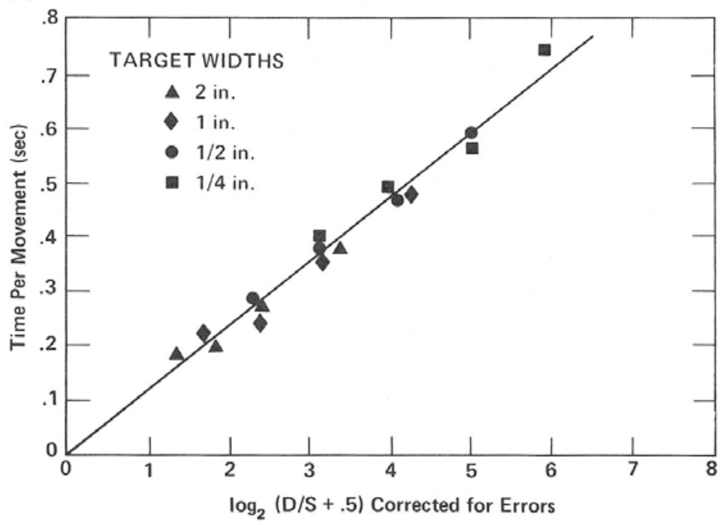
Considers Distance and Target Size

$$T = a + b\log_2(D/S + 1)$$



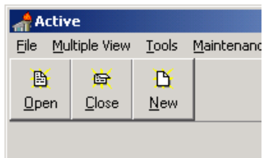
Larger ID → Harder

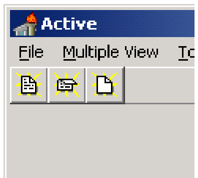
Experimental Data



Toolbar Example

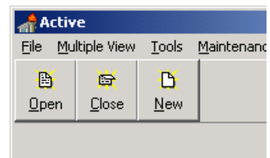
Microsoft Toolbars offer the user the option of displaying a label below each tool. Name at least one reason why labeled tools can be accessed faster. (Assume, for this, that the user knows the tool.)



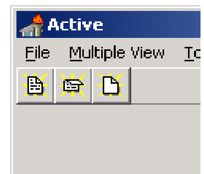


Toolbar Example

1. The label becomes part of the target. The target is therefore bigger. Bigger targets, all else being equal, can always be acccessed faster, by Fitt's Law



2. When labels are not used, the tool icons crowd together



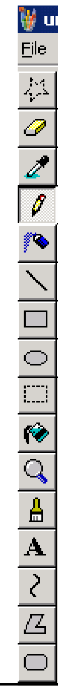
Tool Matrix Example

You have a palette of tools in a graphics application that consists of a matrix of 16x16-pixel icons laid out as a 2x8 array that lies along the left-hand edge of the screen. Without moving the array from the left-hand side of the screen or changing the size of the icons, what steps can you take to decrease the time necessary to access the average tool?

Tool Matrix Example

1. Change the array to 1x16, so all the tools lie along the edge of the screen.

2. Ensure that the user can click on the very first row of pixels along the edge of the screen to select a tool. There should be no buffer zone.



GOMS and KLM

GOMS (Card et al.)

Describe the user behavior in term of

- Goals
  - Edit manuscript, locate line
- Operators
  - Elementary perceptual, motor or cognitive acts
- Methods
  - Procedure for using operators to accomplish goals
- Selection rules
  - Used if several methods are available for a given goal

Family of methods

- KLM, CMN-GOMS, NGOMSL, CPM-GOMS

Quick Example

Goal (the big picture)

- Go from hotel to the airport

~~Methods~~ Operators (or subgoals)?

- Walk, take bus, take taxi, rent car, take train

~~Operators~~ Methods (or specific actions)

- locate bus stop; wait for bus; get on the bus;...

Selection rules (choosing among methods)?

- Example: Walking is cheaper, but tiring and slow
- Example: Taking a bus is complicated abroad

GOMS Output

Execution time
- Add up times from operators
- Assumes **experts** (mastered the tasks)
- **Error free behavior**
- Very good rank ordering
- Absolute accuracy ~10-20%

Procedure learning time (NGOMSL only)
- Accurate for relative comparison only
- Doesn't include time for learning domain knowledge

Using GOMS Analysis

Check that frequent goals can be achieved quickly

Making operator hierarchy is often the value
- Functionality coverage & consistency
  - Does UI contain needed functions?
  - Consistency: are similar tasks performed similarly?
- Operator sequence
  - In what order are individual operations done?

How to do GOMS Analysis

Generate task description

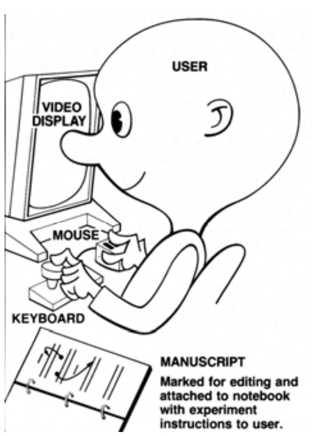
- Pick high-level user **Goal**
- Write **Methods** for reaching Goal - may invoke subgoals
- Write **Methods** for subgoals
  - This is recursive
  - Stops when **Operators** are reached

Evaluate description of task

Apply results to UI

Iterate!

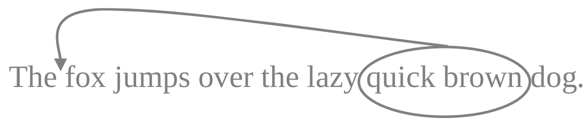
Detailed Task Description



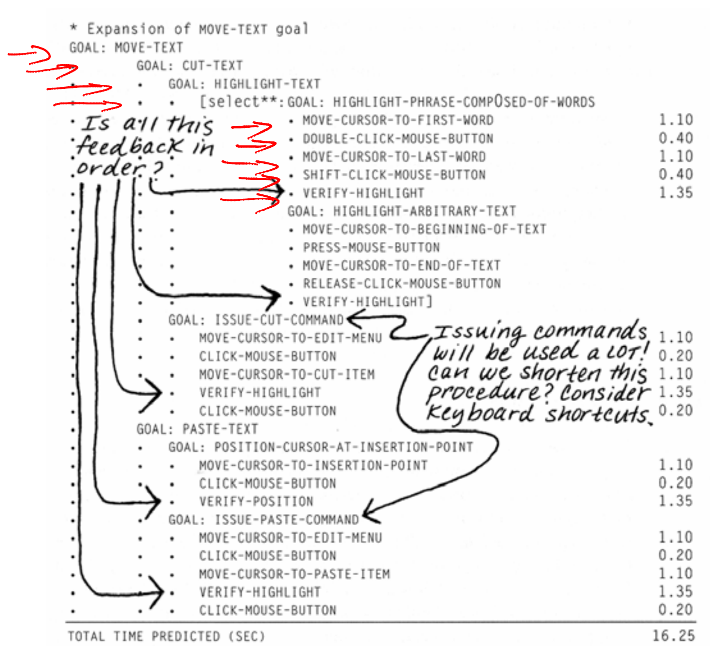

```
GOAL: EDIT-MANUSCRIPT
.  GOAL: EDIT-UNIT-TASK           repeat until no more unit tasks
.  .  GOAL: ACQUIRE-UNIT-TASK
.  .  .  GET-NEXT-PAGE            if at end of manuscript page
.  .  .  GET-NEXT-TASK
.  .  GOAL: EXECUTE-UNIT-TASK
.  .  .  GOAL: LOCATE-LINE
.  .  .  .  [select:  USE-QS-METHOD
                      USE-LF-METHOD]
.  .  .  GOAL: MODIFY-TEXT
.  .  .  .  [select:  USE-S-COMMAND
                      USE-M-COMMAND]
.  .  .  .  VERIFY-EDIT .
```

GOMS Example II

Using a text editor edit the following text as shown



- Goals and sub-goals?
- Operators?
- Methods?
- Selection rules?

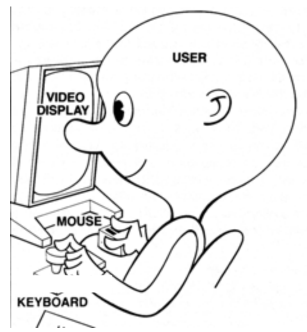


```
* Expansion of MOVE-TEXT goal
GOAL: MOVE-TEXT
.         GOAL: CUT-TEXT
.         .   GOAL: HIGHLIGHT-TEXT
.         .   .   [select**:GOAL: HIGHLIGHT-PHRASE-COMPOSED-OF-WORDS
.         .   .              . MOVE-CURSOR-TO-FIRST-WORD          1.10
.         .   .              . DOUBLE-CLICK-MOUSE-BUTTON          0.40
.         .   .              . MOVE-CURSOR-TO-LAST-WORD           1.10
.         .   .              . SHIFT-CLICK-MOUSE-BUTTON           0.40
.         .   .              . VERIFY-HIGHLIGHT                   1.35
.         .   .             GOAL: HIGHLIGHT-ARBITRARY-TEXT
.         .   .              . MOVE-CURSOR-TO-BEGINNING-OF-TEXT
.         .   .              . PRESS-MOUSE-BUTTON
.         .   .              . MOVE-CURSOR-TO-END-OF-TEXT
.         .   .              . RELEASE-CLICK-MOUSE-BUTTON
.         .   .              . VERIFY-HIGHLIGHT]
.         .   GOAL: ISSUE-CUT-COMMAND
.         .   .   MOVE-CURSOR-TO-EDIT-MENU                        1.10
.         .   .   CLICK-MOUSE-BUTTON                              0.20
.         .   .   MOVE-CURSOR-TO-CUT-ITEM                         1.10
.         .   .   VERIFY-HIGHLIGHT                                1.35
.         .   .   CLICK-MOUSE-BUTTON                              0.20
.         GOAL: PASTE-TEXT
.         .   GOAL: POSITION-CURSOR-AT-INSERTION-POINT
.         .   .   MOVE-CURSOR-TO-INSERTION-POINT                  1.10
.         .   .   CLICK-MOUSE-BUTTON                              0.20
.         .   .   VERIFY-POSITION                                 1.35
.         .   GOAL: ISSUE-PASTE-COMMAND
.         .   .   MOVE-CURSOR-TO-EDIT-MENU                        1.10
.         .   .   CLICK-MOUSE-BUTTON                              0.20
.         .   .   MOVE-CURSOR-TO-PASTE-ITEM                       1.10
.         .   .   VERIFY-HIGHLIGHT                                1.35
.         .   .   CLICK-MOUSE-BUTTON                              0.20
-------------------------------------------------------------------------
TOTAL TIME PREDICTED (SEC)                                        16.25
```

Keystroke Level Model (KLM)

Describe the task using the following operators:

- K: pressing a key or a pressing (or releasing) a button
  t_K = 0.08 - 1.2s (0.2 good rule of thumb)
- P: pointing
  t_P = 1.1s (without button press)
- H: Homing (switching device)
  t_H = 0.4s
- D(*n*,*l*): Drawing segmented lines
  $t_D = 0.9*n + .16*l$
- M: Mentally prepare
  t_M = 1.35s
- R(*t*): system response time
  t_R = t



KLM Heuristic Rules (Raskin's)

0: Insert M
- In front of all K
- In front of all P's selecting a command (not in front of P's ending command)

1: Remove M between *fully anticipated* operators
- PMK → PK

2: if a string of MKs belong to *cognitive unit* delete all M but first
- 4564.23: MKMKMKMKMKMKMK → MKKKKKKK

3: if K is a *redundant terminator* then delete M in front of it
- ↵↵: MKMK → MKK

4a: if K terminates a constant string (command name) <u>delete</u> the M in front of it
- cd↵: MKKMK → MKKK

4b: if K terminates a variable string (parameter) <u>keep</u> the M in front of it
- cd class↵: MKKKMKKKKKMK → MKKKMKKKKKMK

Using KLM

Encode using all physical operator (K, P, H, D(*n*,*l*), R(*t*))

Apply Raskin's KLM rules [0-4]

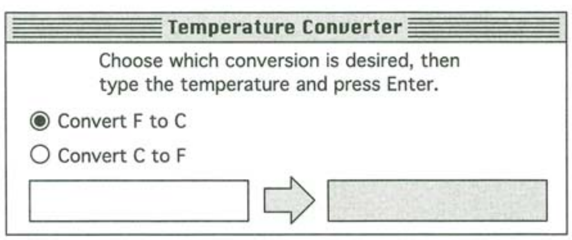
Transform R followed by an M

- If $t \leq t_M$: R(t) → R(0)
- If $t_M < t$: R(t) → R(t - t_M)

Compute the total time by simply adding all time

Converting Temp. Design I

Convert 92.5

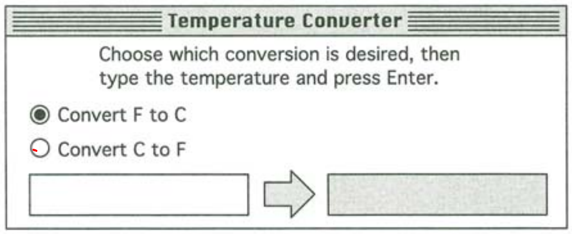


Assume the focus is on the dialog box, so typing on the keyboard will enter text in the text field directly

92.5
K K K K K
MK MK MK MK MK → MKKKKMK

Converting Temp. Design I

Convert 92.5



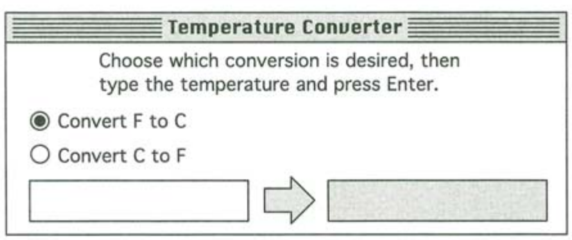
Assume the focus is on the dialog box, so typing on the keyboard will enter text in the text field directly

HPK HKKKK K

→ HMPKHMKKKKMK

Converting Temp. Design I

Convert 92.5



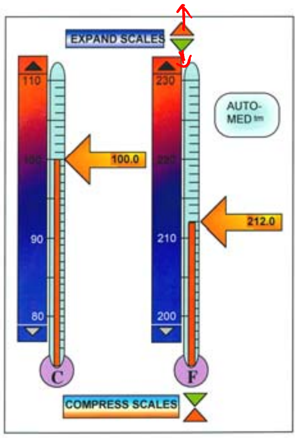
Assume the focus is on the dialog box, so typing on the keyboard will enter text in the text field directly

MKKKKMK (3.7s)

HMPKHMKKKKMK (7.15s)

Average: 5.4s

Converting Temp. Design 2



HPKPK

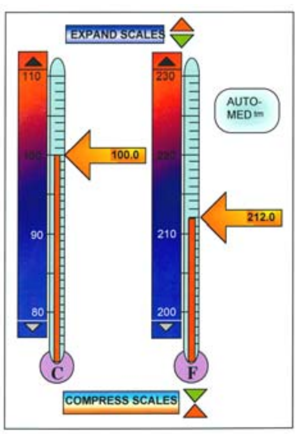
HMPMK MPMK

PMK → PK

HMPK ~~M~~PK

HMPKPK

Converting Temp. Design 2

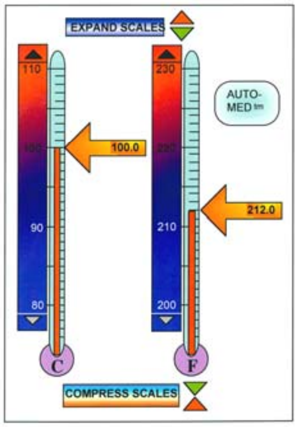


H + 3(PKSK) + PKPK

↓

H + 3(MPKSK) + MPKPK

Converting Temp. Design 2



HMPKPK (4.35s)

H + 3(MPKSK) + MPKPK (21.9s)

Average: 13.125s

Converting Temp. Design 3

Simple text interface with the following prompt:

"To convert temperatures, type the numeric temperature, followed by C if it is in degrees Celsius or F it is in degrees Fahrenheit. Then press enter key. The converted temperature will be displayed"

92.5

K K K K K K

Converting Temp. Design 3

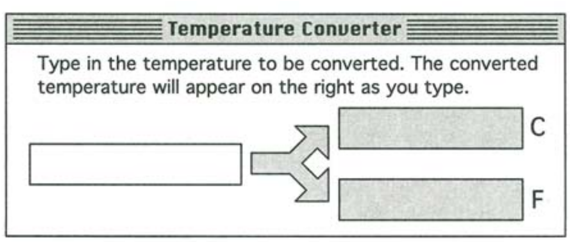
Simple text interface with the following prompt:

"To convert temperatures, type the numeric temperature, followed by C if it is in degrees Celsius or F it is in degrees Fahrenheit. Then press enter key. The converted temperature will be displayed"

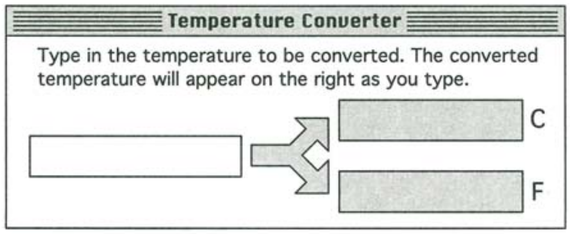
MKKKKKMK (3.9s)

Average: 3.9s

Converting Temp. Design 4



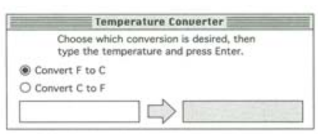
Converting Temp. Design 4

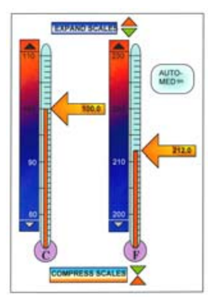


MKKKK (2.15s)

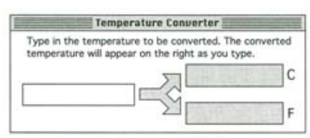
Average: 2.15s

Pros and Cons





"To convert temperatures, type the numeric temperature, followed by C if it is in degrees Celsius or F it is in degrees Fahrenheit. The converted temperature will be displayed"



Applications of GOMS

- Compare different UI designs
- Profiling (time)
- Building a help system? Why?
  - Modeling makes user tasks & goals explicit
  - Can suggest questions users will ask & the answers

What GOMS Can Model

Task must be goal-directed

- Some activities are more goal-directed
  - Creative activities may not be as goal-directed

Task must be a routine cognitive skill

- As opposed to problem solving
- Good for things like machine operators

Serial & parallel tasks (CPM-GOMS)

Advantages of GOMS

- Gives qualitative & quantitative measures
- Model explains the results
- Less work than user study – no users!
- Easy to modify when UI is revised

Research: Need tools to aid modeling process since it can still be tedious

Disadvantages of GOMS

- Not as easy as other evaluation methods
  - Heuristic evaluation, guidelines, etc.
- Takes lots of time, skill, & effort
- Only works for goal-directed tasks
- Assumes tasks **expert** performance without **error**
- Does not address several UI issues,
  - readability, memorizability of icons, commands

Summary

Decision Making and Learning
- Time to make decisions depends on number of options
  - Choosing a movie at Blockbuster
- Learning follows a power law
  - You get faster as you practice

Fitts' Law
- Models movement time to select target
- Time depends on distance and size of target

GOMS and KLM
- A simple model for evaluating interface
- Requires detailed initial task description
- Description may be more useful than perf. predictions

Next Time

Qualitative Evaluation

- Heuristic Evaluation. Nielsen. (Read the 5 linked pages under the heading: **Jakob Nielsen's Online Writings on Heuristic Evaluation**).
- Evaluating the Design Without Users. *Task-Centered User Interface Design.* Chap 4. Lewis and Rieman.