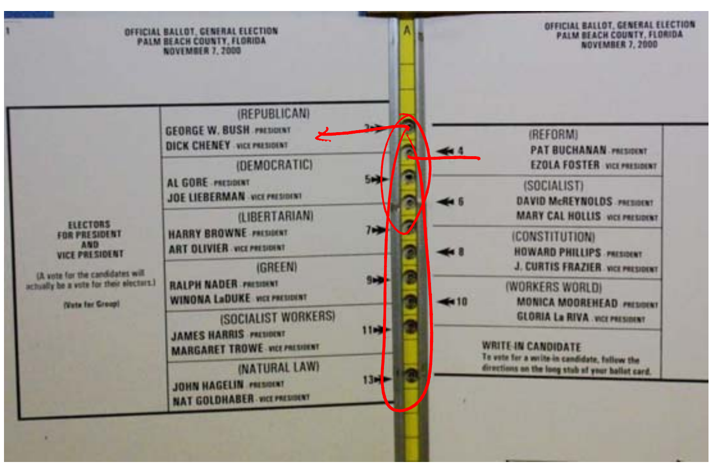
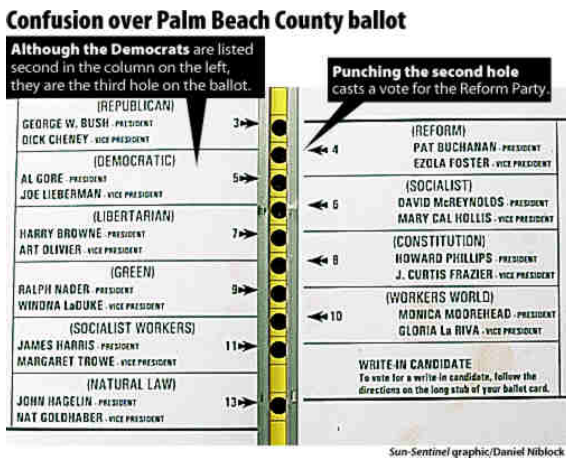
Human Information Processing (KLM, GOMS, Fitts', Hick's)

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

Individual Programming Assignment
(due today)

Project Management/To-Do List

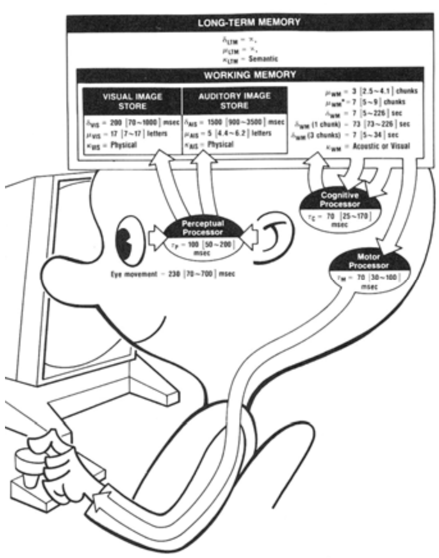
- Tasks have the following properties:
  - Task Name
  - Percentage Completed (0-100%)
  - Start and End date
  - Priority
  - List of people assigned to the task
  - URL related to the task
- Checklist view
  - Include checkbox to automatically set completion percentage to 100%
  - You should be able to see the completion percentage
- Timeline view
- Magic lens: http://dohistory.org/diary/exercises/lens/index.html

LoFi Prototype (due Mar 16)

Low-Fidelity Prototype

- Identify project mission statement
- Create low-fidelity prototype that supports 3 tasks
  - 1 easy, 1 moderate, 1 difficult task
- Create a video prototype showing (cameras next class)
  - How it supports the 3 tasks
  - Context in which is will be used (back story)
- Test the prototype with target users
  - No one from this class
  - Not your friends

Review: Human Info Processor



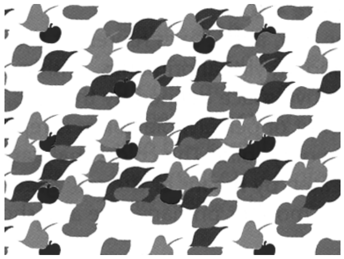
5 Parts

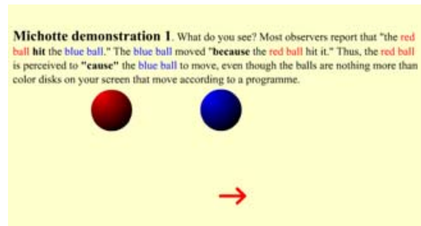
- Perceptual
- Cognitive
- Motor (will discuss today)
- Working memory
- Long-term memory

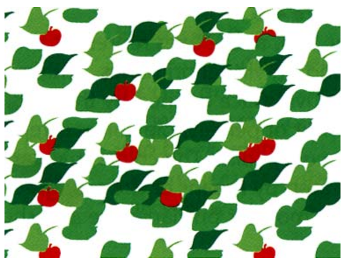
Unified model

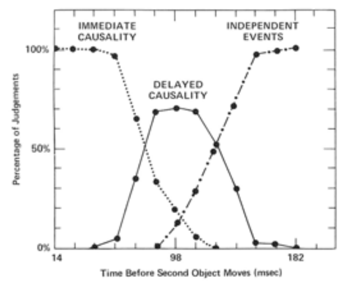
- Probably inaccurate
- Predicts perf. well
- Very influential

Review: Pop-Out and Causality









Topics

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

Memory

Simple Experiment

Volunteer

Start saying **colors** you see in list of words
- When slide comes up
- As fast as you can

Say “done” when finished

Everyone else time it…

Paper
Home
Back
Schedule
Page
Change

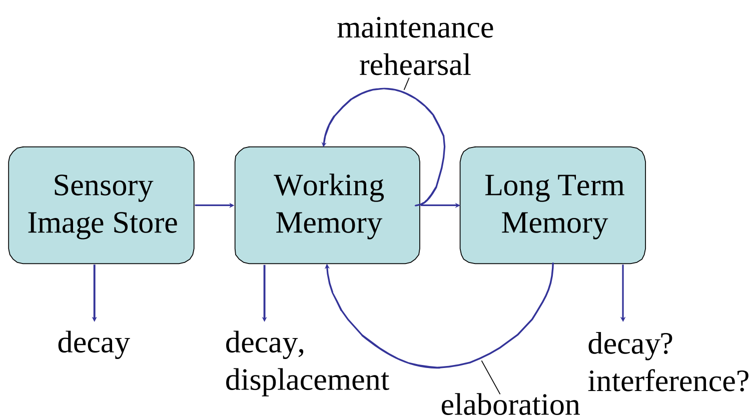
Simple Experiment

Do it again

Say "done" when finished

Blue
Red
Black
White
Green
Yellow

Stage Theory



Stage Theory

Working memory is small
- Temporary storage
  - decay
  - displacement

Maintenance rehearsal
- Rote repetition
- Not enough to learn information well

LTM and Elaboration

Recodes information

Organize (chunking)

Relate new material to already learned material

Link to existing knowledge, categories

Attach meaning
- Make a story

LTM Forgetting

Causes for not remembering an item?

1) Never stored: encoding failure
2) Gone from storage: storage failure
3) Can't get out of storage: retrieval failure

Interference model of forgetting

- One item reduces ability to retrieve another
- Proactive interference (3)
  - Earlier learning reduces ability to retrieve later info.
- Retroactive interference (3 & 2)
  - Later learning reduces the ability to retrieve earlier info.

Recognition over Recall

Recall

- Info reproduced from memory

Recognition

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

We want to design UIs that rely on recognition!

Recall

Write names of the 7 dwarves in Snow White?

Recognition

- Grouchy
- Sneezy
- Smiley
- Sleepy
- Pop
- Grumpy
- Cheerful
- Dopey
- Bashful
- Wheezy
- Doc
- Lazy
- Happy
- Nifty

Recognition

- Grouchy
- **Sneezy**
- Smiley
- **Sleepy**
- Pop
- **Grumpy**
- Cheerful
- **Dopey**
- **Bashful**
- Wheezy
- **Doc**
- Lazy
- **Happy**
- Nifty

Facilitating Retrieval: Cues

Any stimulus that improves retrieval

- – Example: giving hints
- – Other examples in software?
  - icons, labels, menu names, etc.

Anything related to

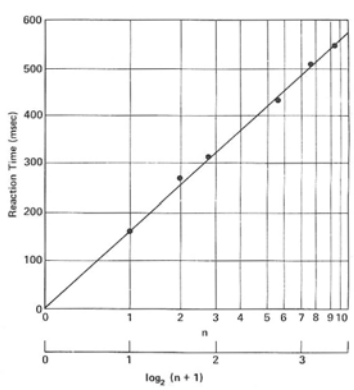
- – Item or situation where it was learned

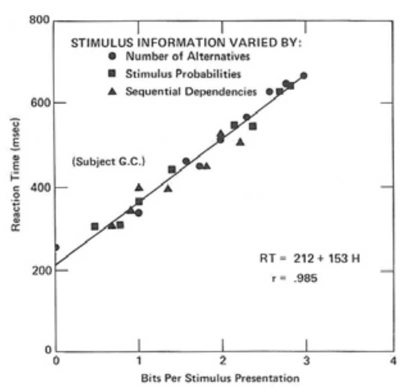
Decision Making and Learning

Hick's Law

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

n = number of choices





Power Law of Practice

- Task time on the n^{th} trial follows a power law

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

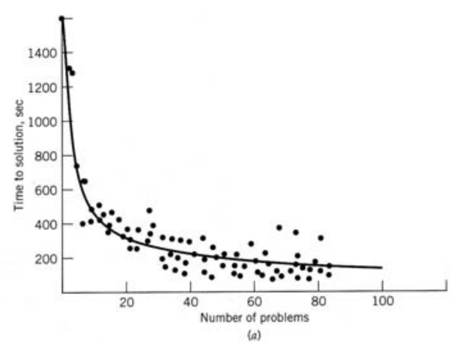
where a = .4, c = limiting constant

Power Law of Practice

- Task time on the n^{th} 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!



Power Law of Practice

- Task time on the nth trial follows a power law

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

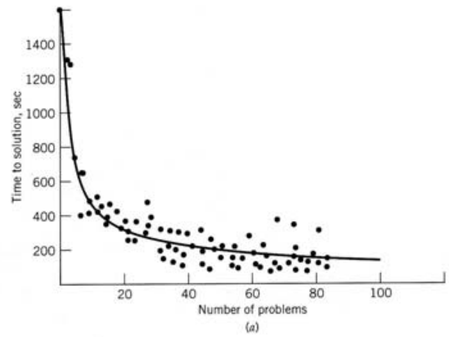
where a = .4, c = limiting constant

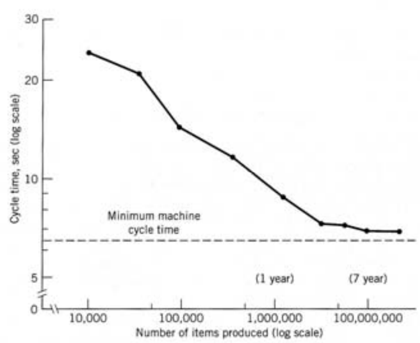
  – You get faster the more times you do it!

Applies to skilled behavior (sensory & motor)

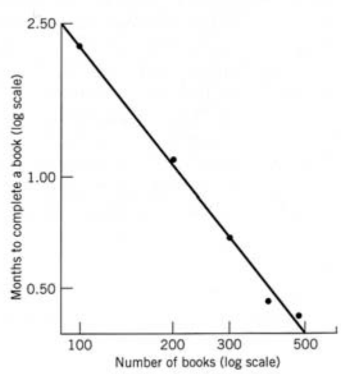
Does not apply to

  – Knowledge acquisition
  – Improving quality





Manual skills



Writing books

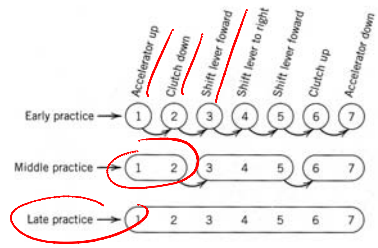
Stages of skill acquisition

Example: Using a manual transmission

Cognitive
- Verbal representation of knowledge

Associative
- Proceduralization
  - Form of chunking

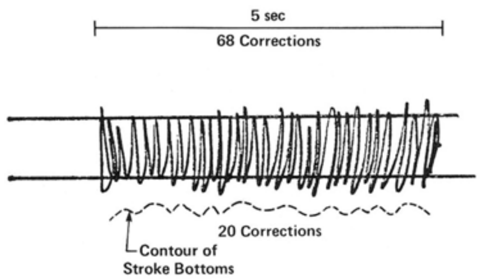
Autonomous
- More and more automated
- Faster and faster
- No cognitive involvement
  - Difficult to describe what to do

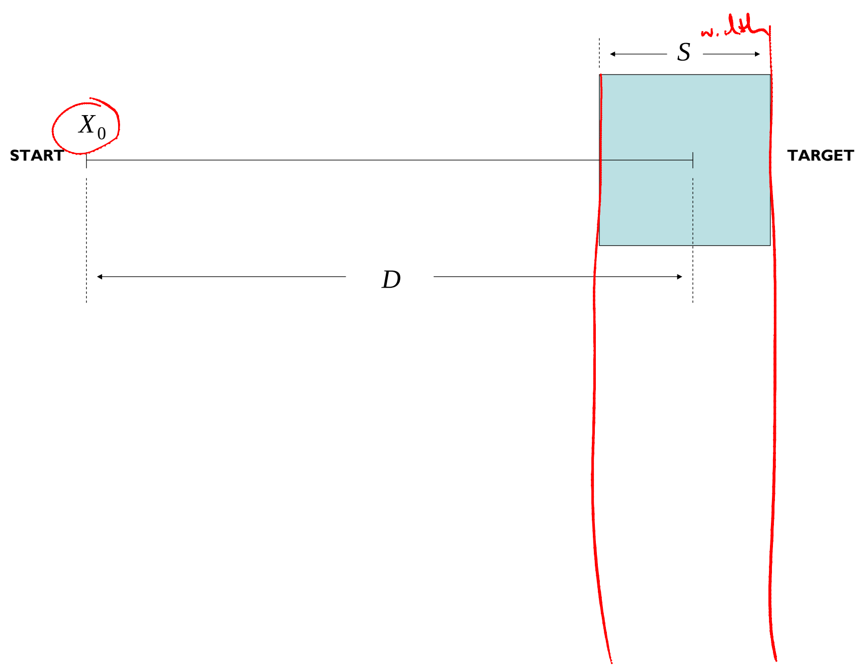


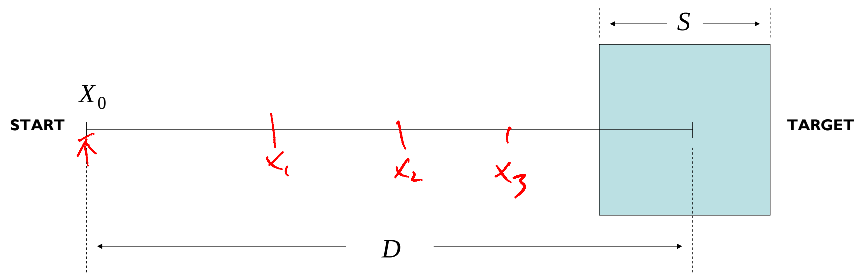
Fitts' Law

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 ➔ $\frac{X_i}{X_{i-1}} = \varepsilon$

$x_1 = \varepsilon x_0 = \varepsilon D$

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

$x_n = \varepsilon^n D$

$\varepsilon^n D \le \frac{S}{2}$ $\quad n = \log_\varepsilon\left(\frac{S}{2D}\right)$ = In $\log_2\left(\frac{2D}{S}\right)$

Fitts' Law

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

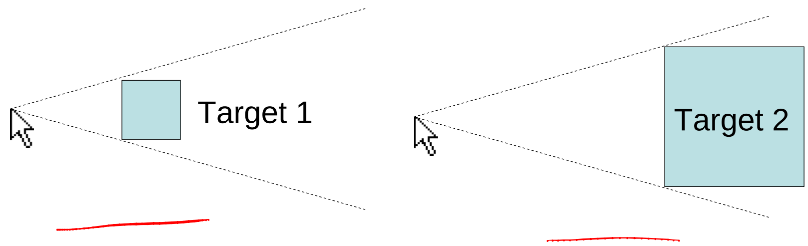
a, b	= constants (empirically derived)
D	= distance
S	= size

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

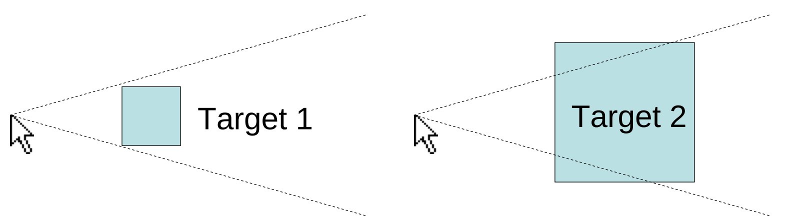
$$T = a + b\log_2(D/S + 1)$$



Same ID → Same Difficulty

Considers Distance and Target Size

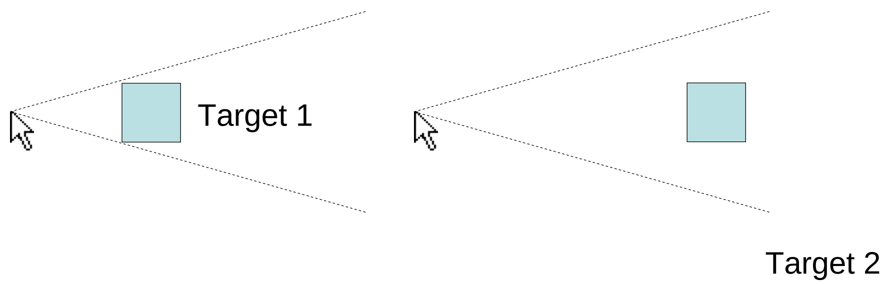
$$T = a + b\log_2(D/S + 1)$$



Smaller ID → Easier

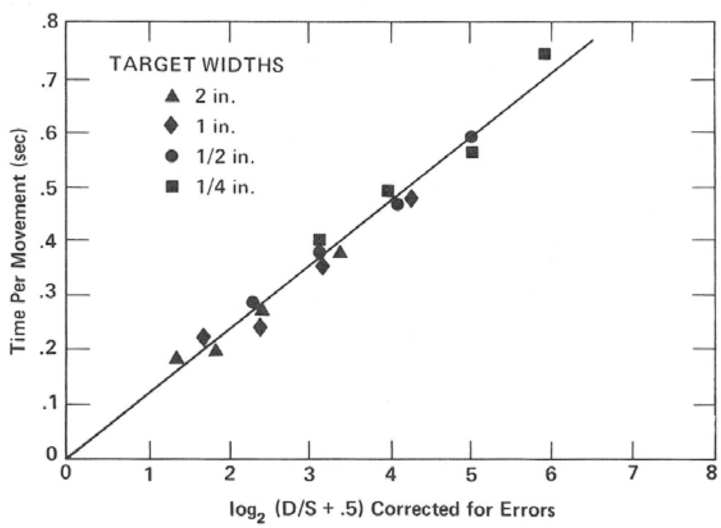
Considers Distance and Target Size

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

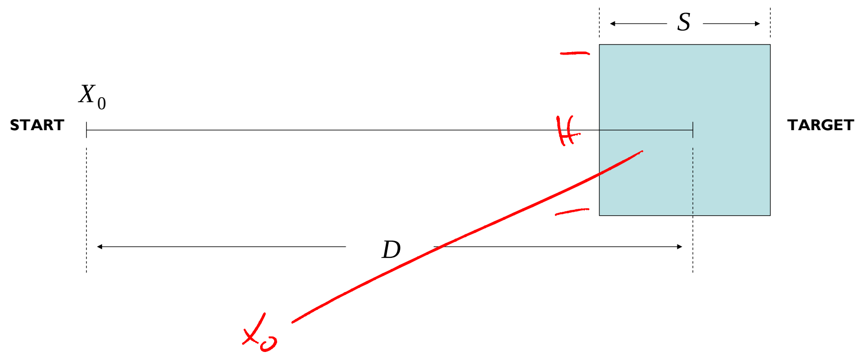


Larger ID → Harder

Experimental Data

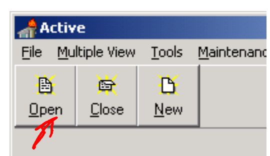


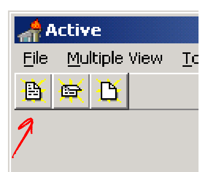
Extend Fitts' Law to 2D Targets?



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.)

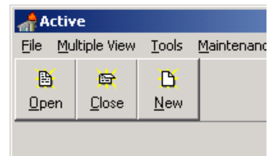


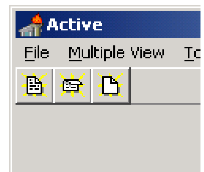


Toolbar Example

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 Fitts' Law

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





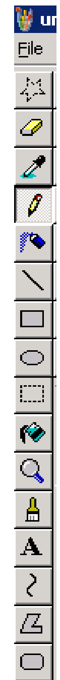
Tool Matrix Example

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

Operators (or subgoals)?

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

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

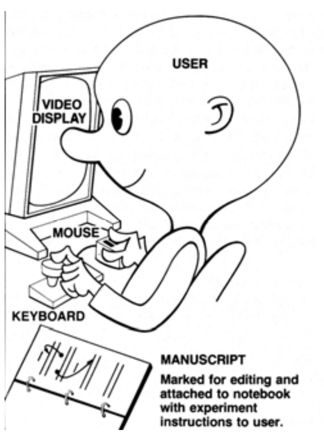
Generate task description
- Pick high-level user **Goal**
- Write **Methods** for reaching Goal - may invoke subgoals
- Write **Methods** for subgoals
  - This is recursive
  - Stops when **Operators** are reached

Evaluate description of task

Apply results to UI

Iterate!

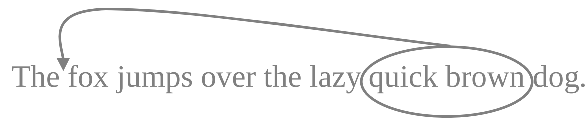
Detailed Task Description



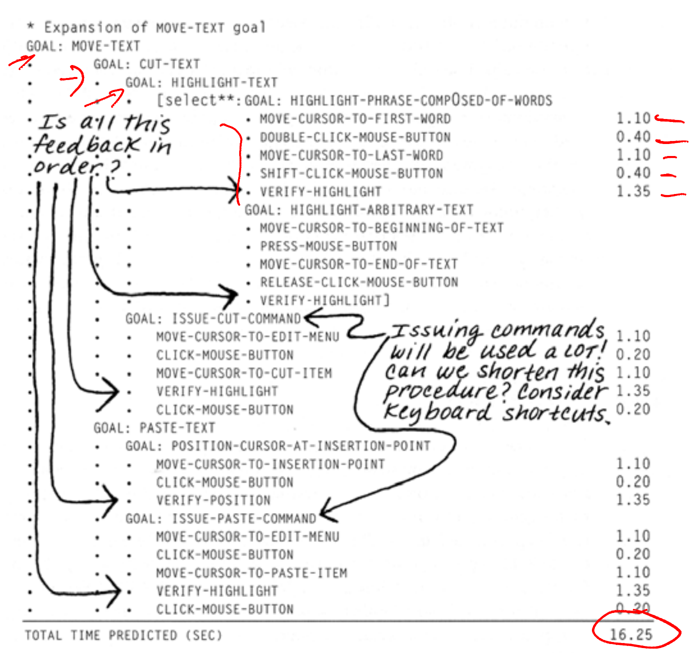

```
GOAL: EDIT-MANUSCRIPT
.   GOAL: EDIT-UNIT-TASK          repeat until no more unit tasks
. .   GOAL: ACQUIRE-UNIT-TASK
. . .   GET-NEXT-PAGE             if at end of manuscript page
. . .   GET-NEXT-TASK
. .   GOAL: EXECUTE-UNIT-TASK
. . .   GOAL: LOCATE-LINE
. . . .   [select:  USE-QS-METHOD
                    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?

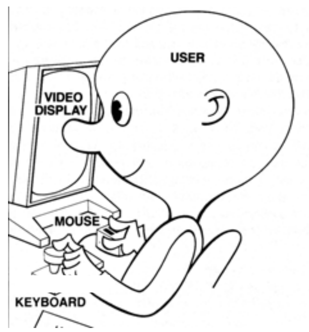


```
* Expansion of MOVE-TEXT goal
GOAL: MOVE-TEXT
.       GOAL: CUT-TEXT
.       .    GOAL: HIGHLIGHT-TEXT
.       .    .    [select**:GOAL: HIGHLIGHT-PHRASE-COMPOSED-OF-WORDS
.       .    .              . MOVE-CURSOR-TO-FIRST-WORD                 1.10
.       .    .              . DOUBLE-CLICK-MOUSE-BUTTON                 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) delete the M in front of it
- cd↵: MKKMK → MKKK

4b: if K terminates a variable string (parameter) keep 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]

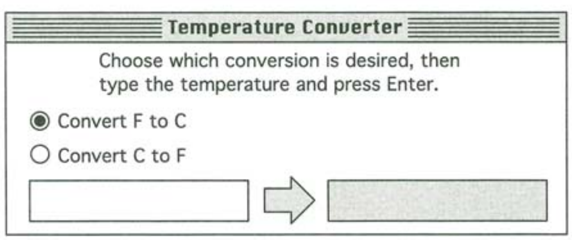
Transform R followed by an M

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

Compute the total time by simply adding all time

Converting Temp. Design I

Convert 92.5

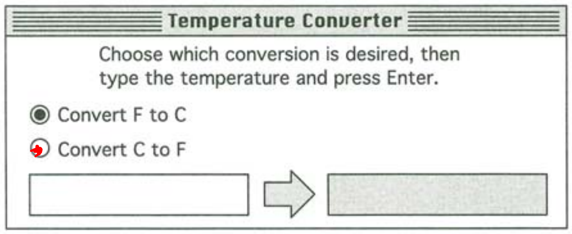


Assume the focus is on the dialog box, so typing on the keyboard will enter text in the text field directly

9 2 . 5 ↵
K K K K K
MK MK MK MK MK = 3.7s

Converting Temp. Design I

Convert 92.5



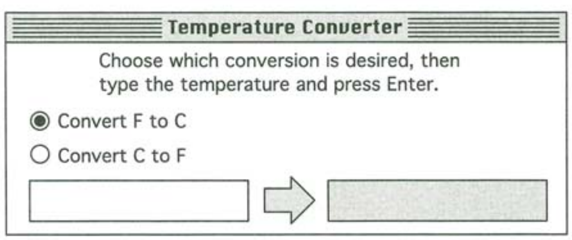
Assume the focus is on the dialog box, so typing on the keyboard will enter text in the text field directly

HPKHKKKKK

HMPKHMKKKKMK = 7.15s

Converting Temp. Design I

Convert 92.5



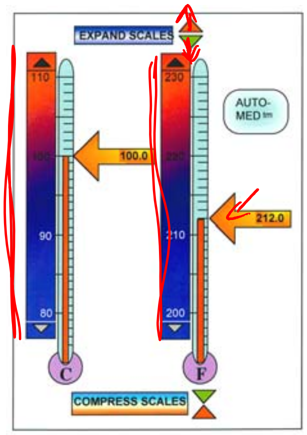
Assume the focus is on the dialog box, so typing on the keyboard will enter text in the text field directly

MKKKKMK (3.7s)

HMPKHMKKKKMK (7.15s)

Average: 5.4s

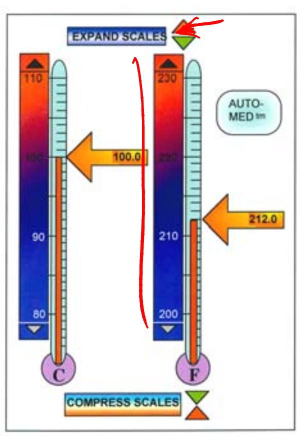
Converting Temp. Design 2



#PKPK

#MPKPK = 4.35s

Converting Temp. Design 2



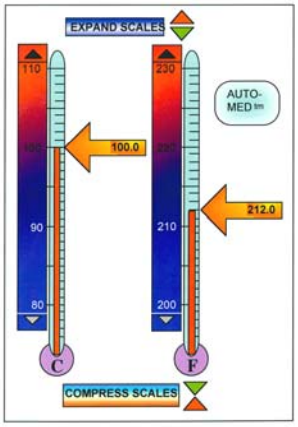
+ 3(PK(8)K) + PKPK

+ PK+3(SK) + PKPK

+ 3(MPKSK) + MPKPK

= 21.9s

Converting Temp. Design 2



HMPKPK (4.35s)

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

Average: 13.125s