

Cognitive Model

By, Dheeba. J/SCOPE

Learning outcomes

- ▶ Measure the response time based on the KLM
- ▶ Identifying goals, operators, methods and selection rules using GOMS.

GOMS

- ▶ Many models make use of a model of mental processing in which the user achieves goals by solving *subgoals in a divide-and-conquer fashion*.
- ▶ The GOMS model of Card, Moran and Newell is an acronym for **G**oals, **O**perators, **M**ethods and **S**election

- **Goals:**
 - Something that the user tries to accomplish (action-object pair, e.g. delete word)
- **Methods:**
 - Well learned sequence of steps that accomplish a task
 - How do you do it on this system? (could be long and tedious...)
- **Selection Rules:**
 - Only when there are clear multiple methods for the same goal.
- **Operators:**
 - Elementary perceptual, cognitive and motor acts that cause change (external vs. mental)
 - Also uses action-object pair (e.g. press key, select menu, make gesture, speak command...)
 - mostly defined by hardware and lower-level software.



Why Do We Use The GOMS Models?

- ▶ the ability to make a priori predictions;
- ▶ the ability to be learned and used by practitioners as well as researchers;
- ▶ coverage of relevant tasks;
- ▶ approximation.



What can GOMS model?

- Task must be goal-directed
 - Some activities are more goal-directed than others
 - Even creative activities contain goal directed tasks
- Task must be a routine cognitive skill
- Can include serial and parallel tasks



GOMS model – types

- ▶ Keystroke-Level Model (KLM)
- ▶ CMN-GOMS (Card Moran Newell)
- ▶ NGOMSL (Natural GOMS language)
- ▶ CPM-GOMS (Critical Path Model)

Keystroke Level Model (KLM)

▶ Why Use the KLM?

- ▶ Consider a task such as “delete a file”
- ▶ Perhaps there are two ways to do the task:
 - ▶ Mouse + menu selection
 - ▶ Keyboard + command entry
- ▶ The KLM can predict the time for each method
- ▶ If used at the design stage, design alternatives may be considered and compared design choices follow



Advantages of Predictive Evaluation

- ▶ Don't have to build UI prototype
 - ▶ Can compare design alternatives with no implementation whatsoever
- ▶ Don't have to test real live users
- ▶ Theory provides explanations of UI problems
 - ▶ So it points to the areas where design can be improved
 - ▶ User testing may only reveal problems, not explain them



How to use KLM to measure and compare the efficiency of user interfaces?

- ▶ Make predictions of the task execution times for a specific design
- ▶ Efficiency is measured as the speed with which a user can accomplish the task.



A KLM Prediction

- ▶ A task is broken into a series of subtasks
- ▶ Total predicted time is the sum of the subtask times:

$$T(\text{execute}) = \sum(\text{time to execute primitive op})$$

- ▶ Primitive operations
 - K key/button press
 - P point to target with mouse
 - H home hands to keyboard or mouse
 - D draw line with mouse
 - M mental preparation (pause)
 - R system response time



-
- ▶ Times for primitive operations are predicated from experiments.
 - ▶ Time to press key/button ranges between
 - 0.08 sec/char fast typist
 - 0.28 sec/char average typist
 - 1.2 sec/char slow typist



Operator	Description	Time (s)
K	<p>PRESS A KEY OR BUTTON</p> <p>Pressing a modifier key (e.g., shift) counts as a separate operation. Time varies with typing skill:</p> <p>Best typist (135 wpm) .08</p> <p>Good typist (90 wpm) .12</p> <p>Average skilled typist (55 wpm) .20</p> <p>Average non-secretary typist (40 wpm) .28</p> <p>Typing random letters .50</p> <p>Typing complex codes .75</p> <p>Worst typist (unfamiliar with keyboard) 1.20</p>	
P	<p>POINT WITH A MOUSE</p> <p>Empirical value based on Fitts' law. Range from .8 to 1.5 seconds. Operator does <i>not</i> include the button click at the end of a pointing operation</p>	1.10
H	HOME HAND(S) ON KEYBOARD OR OTHER DEVICE	.40
$D(n_D, l_D)$	<p>DRAW n_D STRAIGHT-LINE SEGMENTS OF TOTAL LENGTH l_D.</p> <p>Drawing with the mouse constrained to a grid.</p>	$.9 n_D + .16 l_D$
M	MENTALLY PREPARE	1.35
$R(t)$	<p>RESPONSE BY SYSTEM</p> <p>Different commands require different response times. Counted only if the user must wait.</p>	t

KLM Analysis

- ▶ Encode a method as a sequence of physical operators (KPHD)
- ▶ Use heuristic rules to insert mental operators (M)
- ▶ Add up times for each operator to get total time for method



how to create a keystroke level model for a task

- ▶ **First** focus on a particular method for doing the task.
 - ▶ Suppose the task is deleting a word in a text editor. Most text editors offer a variety of methods for doing this, e.g.
 1. click and drag to select the word, then press the Del key
 2. click at the start and shift-click at the end to select the word, then press the Del key
 3. click at the start, then press the Del key N times
 4. double-click the word, then select the Edit/Delete menu command; etc.



-
- ▶ **Second**, encode the method as a sequence of the physical operators: K for keystrokes, B for mouse button presses or releases, P for pointing tasks, H for moving the hand between mouse and keyboard, and D for drawing tasks.
 - ▶ **Third**, insert the mental preparation operators at the appropriate places, before each chunk in the task. Some heuristic rules have been proposed for finding these chunk boundaries.
 - ▶ **finally**, using estimated times for each operator, add up all the times to get the total time to run the whole method.
-



Heuristic Rules for adding M's

- ▶ **Basic idea:**
 - ▶ M before every chunk in the method that must be recalled from long-term memory or that involves a decision
- ▶ **Initiating a task.**
 - ▶ The user has to pause and make a definite decision about what the task is, and what should be done. Thus users often pause before emitting a sequence of actions; this pause should be routinely represented by an M
- ▶ **Making a strategy decision.**
 - ▶ If there is more than one way to proceed, and the decision is not obvious or well practiced, but is important, the user has to stop and think.
- ▶ **Finding something on the screen.**
 - ▶ The user must pause and scan the screen for an item that they do not already know or whose location on the screen they do not already know from practice.
- ▶ **Retrieving a chunk from memory**
 - ▶ Command name
 - ▶ File name
 - ▶ Parameter value



-
- ▶ Verifying that a specification or action is correct.
 - ▶ Before users signal the system to proceed, they often pause and check their entry. For example, users often stop and examine a command before hitting return, or check a dialog box before clicking on the OK, or that a destination folder is reverse-video before releasing the mouse button
 - ▶ Pointing to an object on the screen should be preceded by a mental operator to locate the object.
 - ▶ If something on the screen changes in response to user input, there should be a step to verify that the desired result appeared.
 - ▶ Before every keypress. And if the keys are consecutive then one M operator is placed.
-



Example

- ▶ keystroke-level models for two methods that delete a word.

Method 1:

- ▶ The first method clicks at the start of the word, shift-clicks at the end of the word to highlight it, and then presses the Del key on the keyboard.

Method 2:

- ▶ The second method clicks at the start of the word, then presses Del enough times to delete all the characters in the word.



-
- ▶ **K** - 0.28 s average typist
 - ▶ **B** - 0.1 s
 - ▶ **P** - 1.1 s
 - ▶ **H** - 0.4 s
 - ▶ **M** - 1.2 s



Method 1 - Shift-click selection

- ▶ M
- ▶ P [start of word]
- ▶ BB [click]
- ▶ H
- ▶ M
- ▶ K [shift]
- ▶ M
- ▶ P [end of word]
- ▶ BB [click]
- ▶ H [to keyboard]
- ▶ M
- ▶ K [Del]
- ▶ M

Total: $5M + 2P + 4B + 2K$



Method 2 - pressing Del key n times

- ▶ M
- ▶ P [start of word]
- ▶ BB [click]
- ▶ H
- ▶ M
- ▶ K [Del] $\times n$ [where n = length of word]
- ▶ M

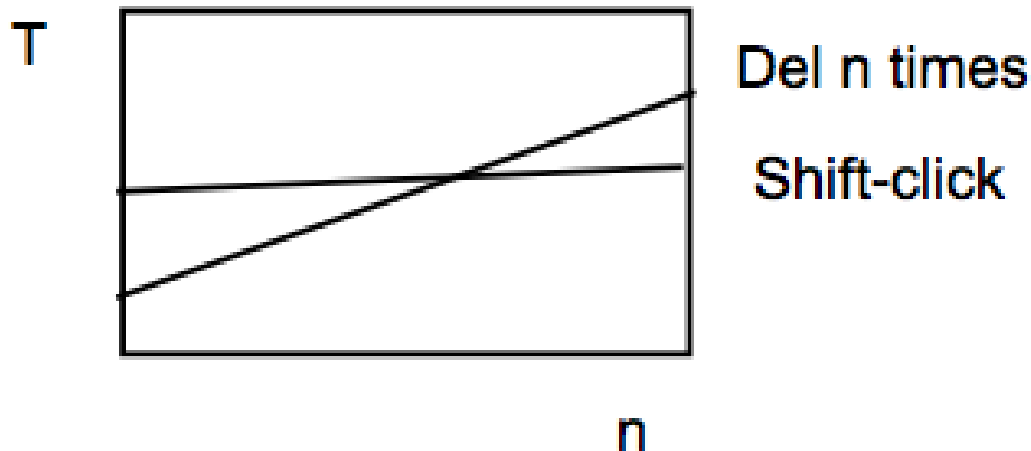
Total: $3M+P+2B+H+nK$



Comparing efficiency of different user interface designs

► **parametric analysis**

- e.g., as we vary the parameter n (the length of the word to be deleted), how do the times for each method vary?



Consider the following three different interfaces for collecting the date of birth. Calculate the time taken to perform each of these operations using a KLM and perform a task execution time analysis to find which interface model is a best alternative for collecting DOB. List the design heuristics that should be considered for placing the M operator for the following interface designs.

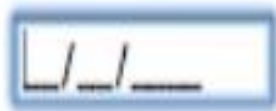


Interface design 1: A form labeled "Birthday:" followed by three separate spinner controls. The first spinner is labeled "Month:", the second "Day:", and the third "Year:". Each spinner has up and down arrows for selection.



Interface design 2: A form labeled "Birthday" followed by three input fields. The first is a text box labeled "Day". The second is a dropdown menu labeled "- Select Month -" with up and down arrows. The third is a text box labeled "Year".

* Date of birth
(dd/mm/yyyy)



Interface design 3: A single text input field for the date of birth, with a blue border and a mask showing the format dd/mm/yyyy (e.g., _/_/_/_).

K - 0.28 s average typist

B - 0.1 s

P - 1.1 s

H - 0.4 s

M - 1.2 s

CMN GOMS

Suppose we want to find out the definition of a word from an online dictionary. How can we model this task with (CMN)GOMS?

- ▶ list the goals (high level tasks) first
 - Goal: Access online dictionary (first, we need to access the dictionary)
 - Goal: Lookup definition (then, we have to find out the definition)



- ▶ Next, we have to determine the methods (operator or goal-operator sequence) to achieve each of these goals
-

Goal: Access online dictionary

Operator: Type URL sequence

Operator: Press Enter

Goal: Lookup definition

Operator: Type word in entry field

Goal: Submit the word

Operator: Move cursor from field to Lookup button

Operator: Select Lookup

Operator: Read output



-
- ▶ The previous example illustrates the concepts of goals and goal hierarchy, operators and methods
 - ▶ The other important concept in **(CMN)GOMS** is the selection rules

Suppose we have a window interface that can be closed in either of the two methods: by selecting the 'close' option from the file menu or by selecting the Ctrl key and the F4 key together. How we can model the task of “closing the window” for this system using (CMN) GOMS?

Goal: Close window

[Select Goal: Use menu method

Operator: Move mouse to file menu

Operator: Pull down file menu

Operator: Click over close option

Goal: Use Ctrl+F4 method

Operator: Press Ctrl and F4 keys together]



Cognitive complexity theory (CCT)

- ▶ Cognitive complexity theory, introduced by *Kieras and Polson*.
- ▶ two parallel descriptions: one of the **user's goals** and the other of the **computer system**.
 - ▶ **user's goals** is based on a GOMS-like goal hierarchy – expressed in the form of *production rules*.
 - ▶ **system grammar**, CCT uses *generalized transition networks*, a form of *state transition network*
- ▶ The production rules are a sequence of rules: (rule program is written in LISP)
 - if condition then action*
- ▶ Where, *condition* is a statement about the contents of working memory. *action* may consist of one or more elementary actions, which may be either changes to the working memory, or external actions such as keystrokes

Example

Selection rule set for goal: Highlight text

If text-is word, then accomplish goal: Highlight word.

If text-is arbitrary, then accomplish goal: Highlight arbitrary text.

Return with goal accomplished.



Delete a file using GOMS

```
GOAL: DELETE-FILE
.   GOAL: SELECT-FILE
.   .   [select:  GOAL: KEYBOARD-TAB-METHOD
.   .   GOAL: MOUSE-METHOD]
.   .   VERIFY-SELECTION
.   GOAL: ISSUE-DELETE-COMMAND
.   .   [select*: GOAL: KEYBOARD-DELETE-METHOD
.   .   PRESS-DELETE
.   .   GOAL: CONFIRM-DELETE
.   .   GOAL: DROP-DOWN-MENU-METHOD
.   .   MOVE-MOUSE-OVER-FILE-ICON
.   .   CLICK-RIGHT-MOUSE-BUTTON
.   .   LOCATE-DELETE-COMMAND
.   .   MOVE-MOUSE-TO-DELETE-COMMAND
.   .   CLICK-LEFT-MOUSE-BUTTON
.   .   GOAL: CONFIRM-DELETE
.   .   GOAL: DRAG-AND-DROP-METHOD
.   .   MOVE-MOUSE-OVER-FILE-ICON
.   .   PRESS-LEFT-MOUSE-BUTTON
.   .   LOCATE-RECYCLING-BIN
.   .   MOVE-MOUSE-TO-RECYCLING-BIN
.   .   RELEASE-LEFT-MOUSE-BUTTON]
```

*Selection rule for GOAL: ISSUE-DELETE-COMMAND

If hands are on keyboard, use KEYBOARD-DELETE-METHOD,
else if Recycle bin is visible, use DRAG-AND-DROP-METHOD,
else use DROP-DOWN-MENU-METHOD

ATM machine – Basic operators

- ▶ **GOAL: GET-MONEY**
 - ▶ **GOAL: USE-CASH-MACHINE**
 - ▶ INSERT-CARD
 - ▶ ENTER-PIN
 - ▶ SELECT-GET-CASH
 - ▶ ENTER-AMOUNT
 - ▶ COLLECT-CARD
 - ▶ COLLECT-MONEY (outer goal satisfied!)

Natural GOMS

- ▶ Natural GOMS language (NGOMSL), developed by Kieras, provides a structured natural-language notation for GOMS analysis and describes the procedures for accomplishing that analysis.
- ▶ NGOMSL provides
 - ▶ A method for measuring the time it will take to learn specific method of operation
 - ▶ A way to determine the consistency of a design's methods of operation

Method for goal: Cut text

Step 1. Accomplish goal: Highlight text.

Step 2. Return that the command is CUT, and accomplish goal: Issue a command.

Step 3. Return with goal accomplished. ...

Selection rule set for goal: Highlight text

If text-is word, then accomplish goal: Highlight word.

If text-is arbitrary, then accomplish goal: Highlight arbitrary text. Return with goal accomplished. ...

Method for goal: Highlight arbitrary text

Step 1. Determine position of beginning of text (1.20 sec)

Step 2. Move cursor to beginning of text (1.10 sec)

Step 3. Click mouse button. (0.20 sec)

Step 4. Move cursor to end of text. (1.10 sec)

Step 5. Shift-click mouse button. (0.48 sec)

Step 6. Verify that correct text is highlighted (1.20 sec)

Step 7. Return with goal accomplished.

This NGOMSL model predicts that it will take 5.28 seconds to highlight arbitrary text

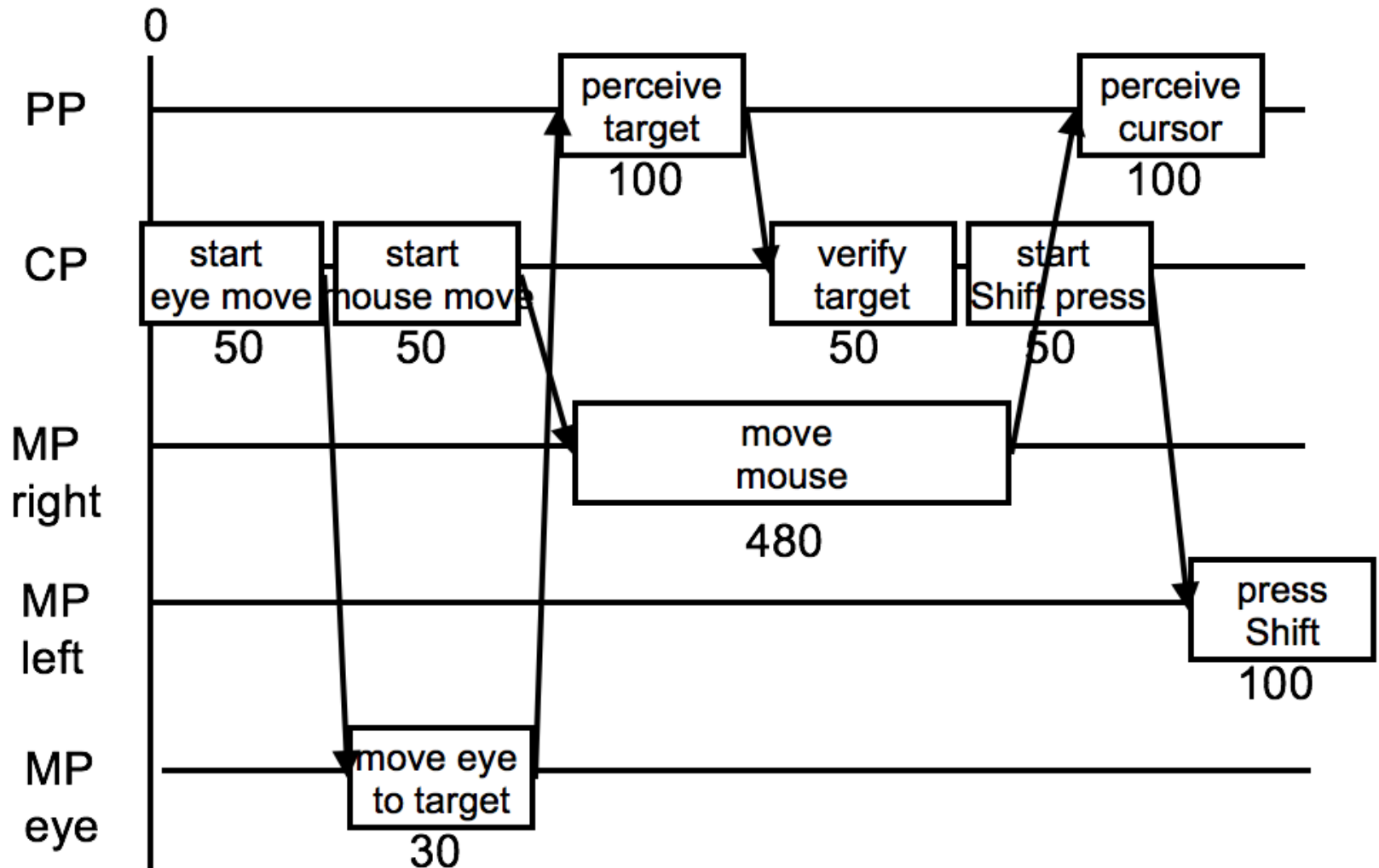


CPM - GOMS

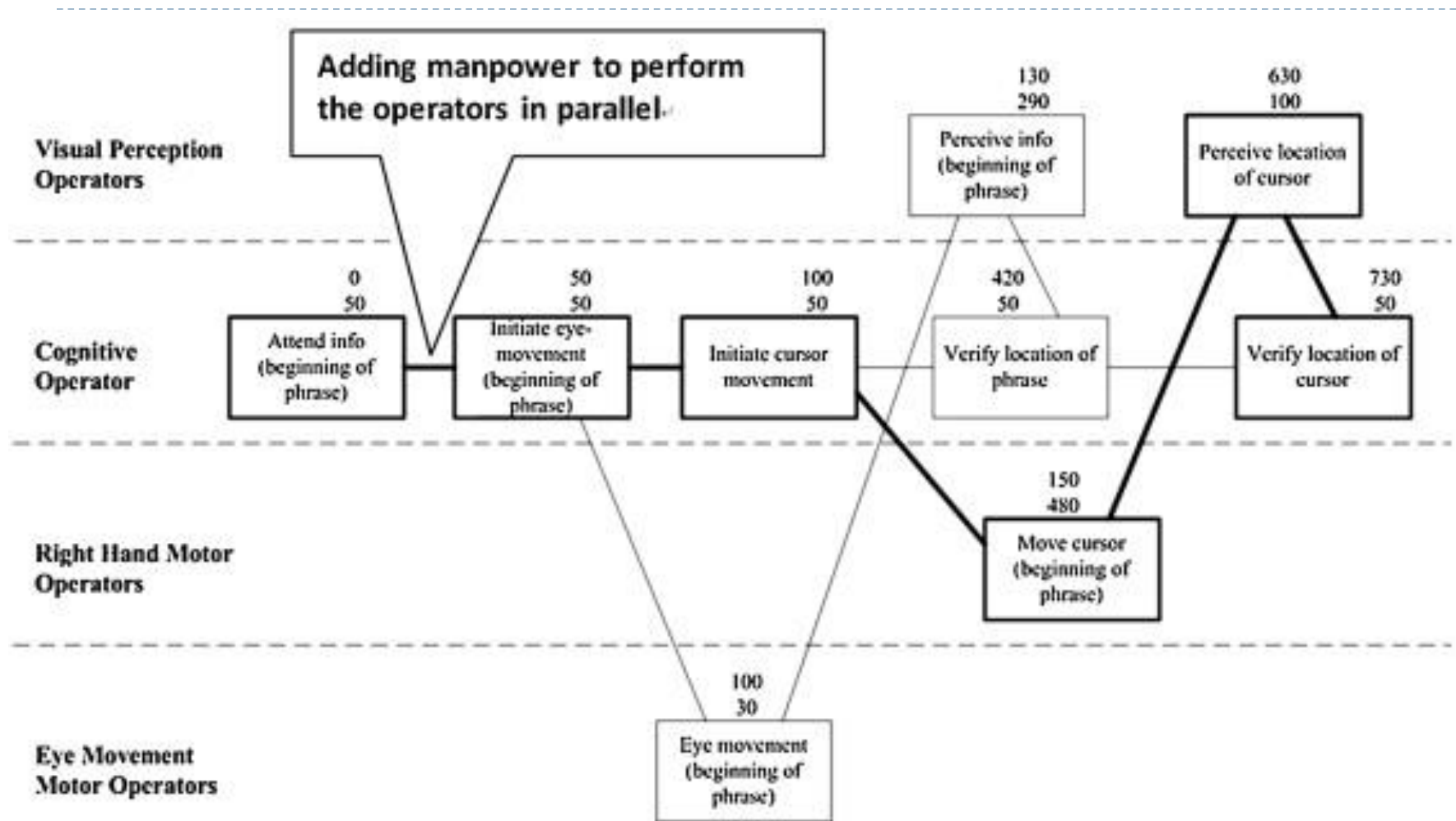
▶ CPM – Critical Path Model

- ▶ CPM-GOMS builds on previous GOMS models by assumed that perceptual, cognitive and motor operators can be performed in parallel
- ▶ It employs a schedule chart (also known as a PERT chart) to represent operators and dependencies between operators.
- ▶ Example, the goal READ-SCREEN, when an eye movement is required.
- ▶ The sequence which produces the longest path through the chart is called the critical path, and it represents an estimate of the total time required to perform the task.

- **Point-Shift-click operation** - the path that takes the longest time, since it will determine the total time for the method.



CPM GOMS model for Move text method



Modelling structure – Hick's law

- ▶ Hick's Law (or the Hick-Hyman Law) is named after a British and an American psychologist team of William Edmund Hick and Ray Hyman
- ▶ Hick's law (Hick-Hyman law) can help to optimize menu structures.
- ▶ Hick's Law states that increasing the number of choices will increase the decision time logarithmically.
- ▶ The reaction time has **a logarithmic curve**, since users can process information presented in categories or groups, therefore when the user makes a choice, he/she eliminates whole groups of other choices — that is why the **curve is not linear**.

- ▶ As a designer, you will use Hick's Law to examine how many functions you should offer at any part of your website and how this will affect your users' overall approach to decision making.

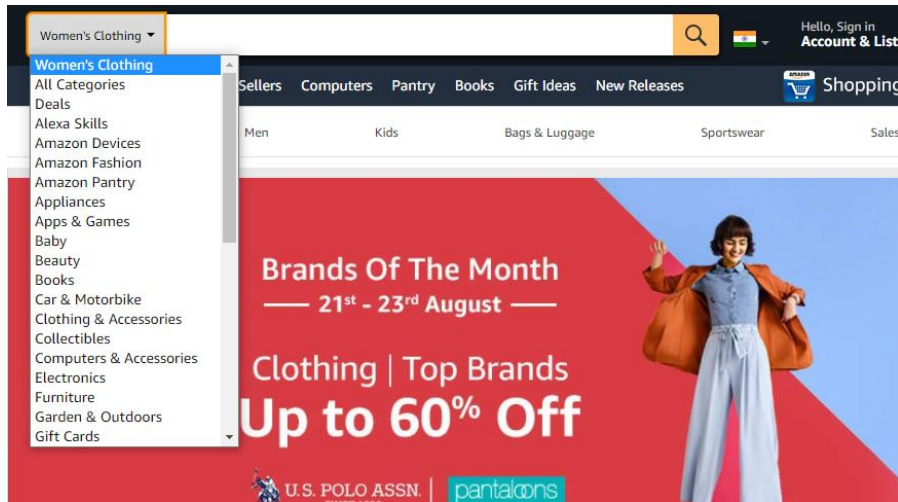
- ▶ The formula for Hick's Law is defined as follows:

$$RT = a + b \log_2 (n + 1)$$

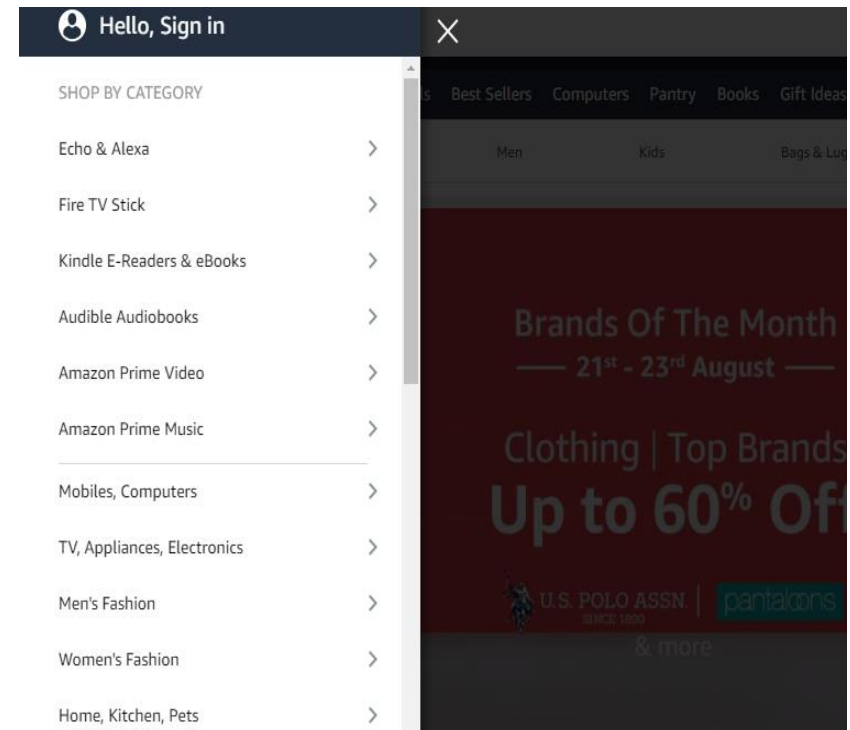
- ▶ The coefficients a and b are arrived at from experiments which depend on the person, training, other factors.
- ▶ If the number of choices is related with probability of occurrences

$$H = \sum P_j \log_2 \left(\frac{1}{P_j} + 1 \right)$$

- ▶ Hick's Law determined the number of controls on your microwave or your washing machine.
- ▶ A design principle known as "K.I.S.S." ("Keep It Short and Simple") became recognized in the 1960s for its effectiveness in this regard.
- ▶ **Categorizing Choice:**
 - ▶ You can see Hick's Law in action in the navigation of almost any website.
 - ▶ If your menus offered direct access to every link within your site, you could quickly overwhelm the visitor.
- ▶ **Obscuring Complexity**
 - ▶ If you have a complex process, you can use Hick's Law to rationalize only presenting specific parts of that process at any one time on the screen.
 - ▶ Instead of throwing the entirety of your payment process up in a long, complex form, you can break it down into prompting users to register their e-mail and create a password.
 - ▶ Then, you can give them another screen with shopping cart details, then another which collects delivery information and so on.



- ▶ Amazon gives you these options to find what you are looking for with ease.
- ▶ Though the options are large it is broken into groups
- ▶ Imagine the amount of options that will be available on amazon if they don't follow the hick's law



Case study – Hick's Law

- ▶ Zomato is one of the biggest online food ordering platforms today. In a nutshell it is the biggest and widest menu card a person has to choose food from. So naturally, the choices are not only ample but each more delicious than the other. Zomato does offer a range of filtering options to narrow down our food choices, such as — delivery time, cost of the food, cuisine, offers and many more. Even with the availability of these filters, users can take up to 30 minutes to decide what to order, which gets more confusing if you are deciding as a group. So how do we apply hick's law here.

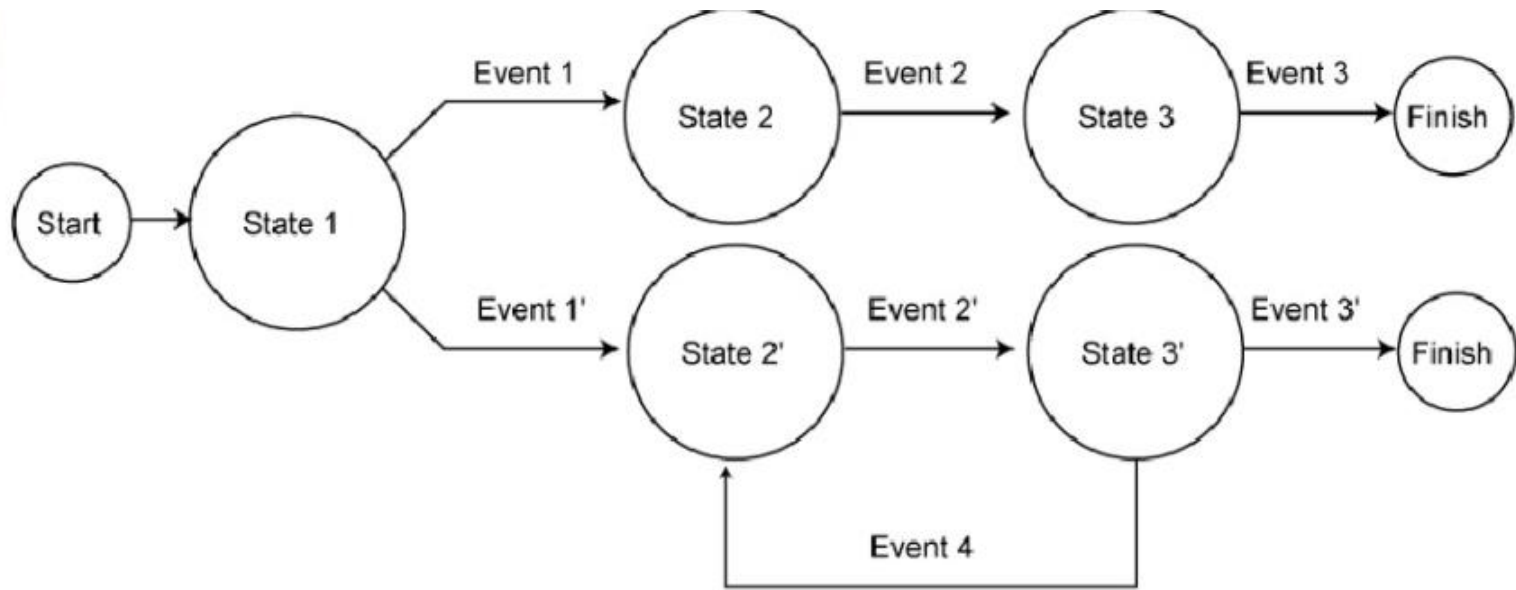
-
- ▶ delicious looking fresh food pictures, attached with every restaurant and their menu cards. The user is constantly browsing through menu items while looking at these mouth-watering images.
 - ▶ So, is it a coincidence that the user is presented with numerous choices, thereby increasing the decision-making time while constantly exposing them to food pictures? If it is, Zomato has cracked the code to smartly getting more orders from one user without having to persuade them any other way. After all, a hungry user is a user they can serve well.
 - ▶ So, the crux here is, choices are not always harmful for a company's profit.

Modelling Dynamics - State Transition Network (STN)

- ▶ Interaction designs involve dynamic feedback loops between the user and the system
 - ▶ User actions alter the state of the system, which in turn influences the user's subsequent actions.
 - ▶ Interaction designers need tools to explore how a system undergoes transition from one state to another.
- ▶ STN helps to explore
 - ▶ Menus
 - ▶ Icons
 - ▶ Tools
- ▶ STN helps to show the operation of the peripheral devices.

-
- ▶ STNs are the most spontaneous, which knows that a dialog fundamentally denotes to a progression from one state of the system to the next.
 - ▶ The syntax of an STN consists of the following two entities
 - ▶ **Circles** – A circle refers to a state of the system, which is branded by giving a name to the state.
 - ▶ **Arcs** – The circles are connected with arcs that refers to the action/event resulting in the transition from the state where the arc initiates, to the state where it ends.
 - ▶ STNs are appropriate for showing sequential operations that may involve choice on the part of the user as well as for expressing iterations.

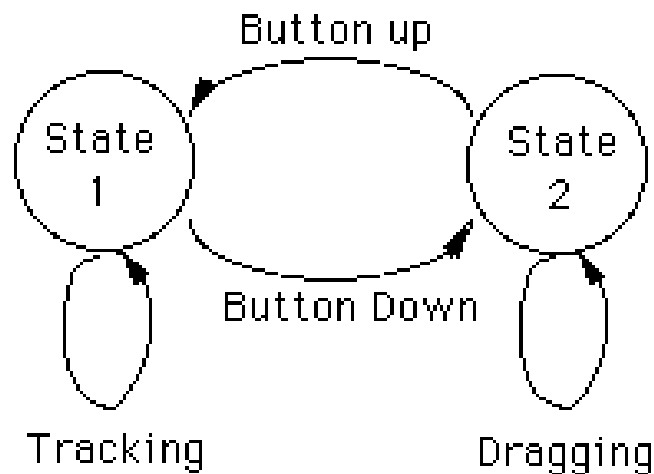
► Structure of STN



Modelling dynamics – three state model

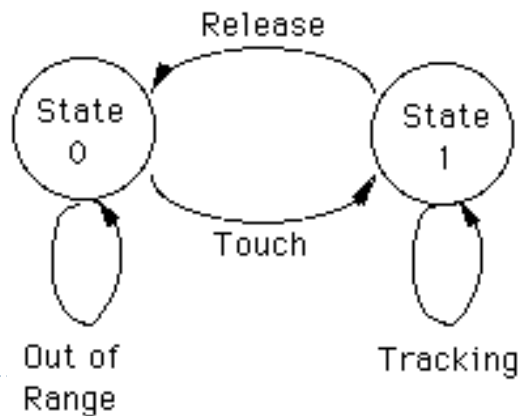
- ▶ The three state model can help designers to determine appropriate I/O devices for specific interaction designs.
- ▶ All input devices are not created equal in their capabilities, nor input techniques (such as pointing, dragging, and rubber-banding) in their demands.
- ▶ Variations are in terms of qualitative (types of signal) and quantitative (amount of signal)

- ▶ Consider moving the mouse without the button pushed.
- ▶ One way to characterize the state of the system at this point is as *tracking*.
- ▶ If, however, we point at an icon, depress the mouse button and move the mouse while holding the button down, we have entered a new state, *dragging*.



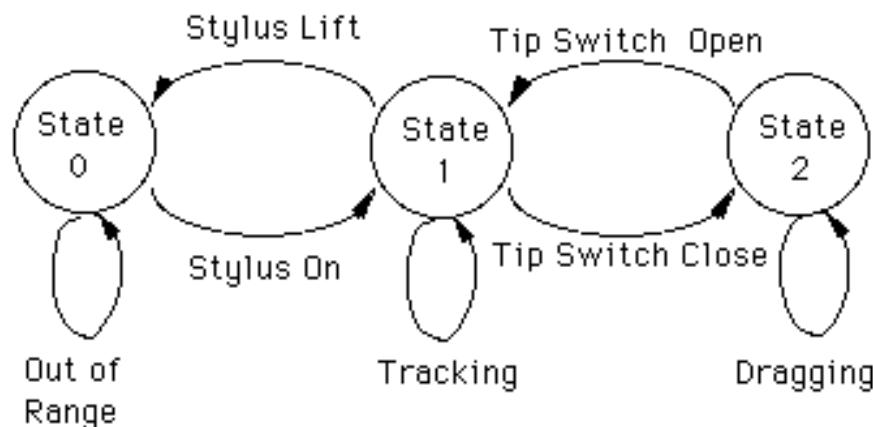
Simple 2-State Transaction

- ▶ Consider now the situation if a **touch tablet** rather than a mouse was connected to the system.
- ▶ For the purpose of the example, let us assume that the touch tablet is capable of sensing only one bit of pressure, namely touch or no-touch.
- ▶ The first state, (State 0), is what we will call *out of range*, (OOR).
- ▶ In this state, any movement of the finger has no effect on the system. It is only when the finger comes in contact with the tablet that we enter the State 1.



State 0-1 Transaction

- ▶ Assume a graphics tablet with stylus.
- ▶ In State 0, the stylus is off of the tablet and the tip switch in its open state.
- ▶ Moving the stylus has no effect since it is out of range (OOR). When the stylus is in range, the tracking symbol follows the stylus' motion (State 1: tracking).
- ▶ **Extra pressure** on the stylus closes the tip switch, thereby moving the system into State 2.

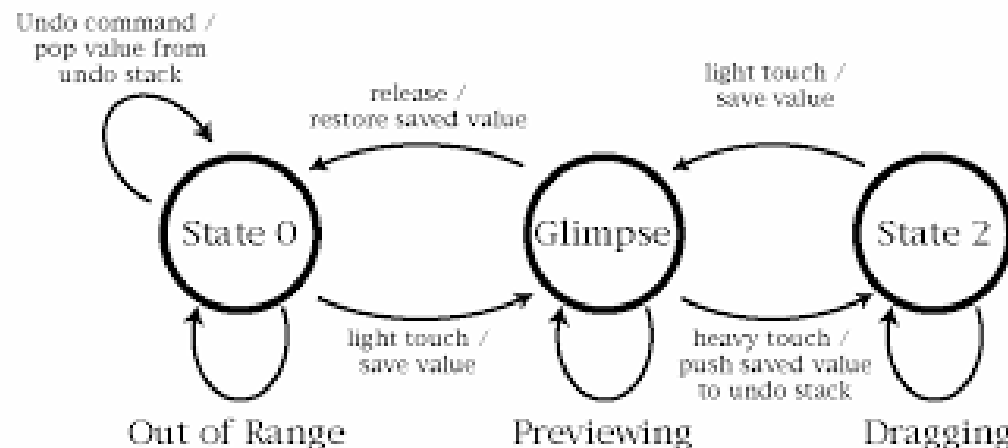


State 0-1-2 Transaction

Modelling dynamics – Glimpse Model

- ▶ Technique that supports the previewing of navigation, exploration, and editing operations by providing convenient **Undo** for unsuccessful and/or undesirable actions **on multi-level input devices** such as touch screens and pen-based computers.
- ▶ By adding a **Glimpse state** to traditional three-state pressure sensitive input devices, users are able to preview the effects of their editing without committing to them.
- ▶ Creativity is enhanced when users are able to easily retract their changes if the result is unsatisfactory.
- ▶ multi-level input devices - input device is capable of sensing at least two levels of input
 - ▶ e.g. a stylus that senses light and heavy pressure or a mouse with a two-state button
- ▶ providing positional feedback
 - ▶ Onscreen or implicit

- ▶ three-state model for pressure sensitive input
- ▶ light pressure input results in the “tracking” of the input device (similar to moving the mouse while its button is up).
- ▶ Heavy pressure input results in “dragging” operations (similar to moving the mouse while its button is down)

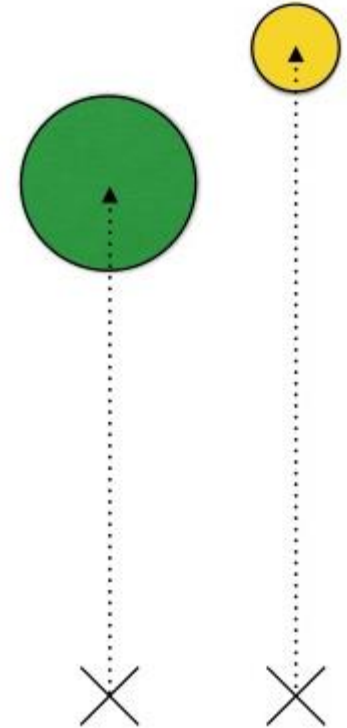


Applications

- Some applications
 - **Pan and zoom interfaces**—Preview different magnification levels
 - **Navigation in a 3D world**—Quick inspection of an object from different perspectives
 - **Color selection in a paint program**—Preview the effects of color manipulation
 - **Volume control**—Preview different volume levels
 - **Window control**—Moving or resizing windows to view occluded objects
 - **Scrollbar manipulation**—Preview other sections of a document

Physical Models – Fitts Law

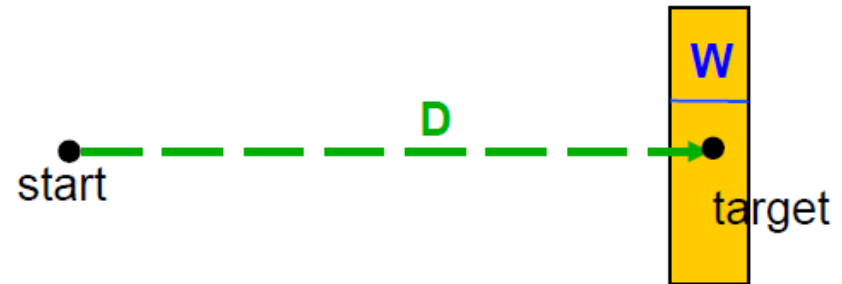
- ▶ named after **Paul Fitts**, for his study of pointing.
- ▶ pointing with mouse, using touch screens and the range of pointing devices for 2D displays.
- ▶ the law tells us **how long it will take to move a pointer from a specific position to hit different targets.**
- ▶ targets that are larger and closer are easier to hit than ones that are smaller and farther away.
- ▶ we can also use Fitts' law to ***compare different input devices.***



Fitts' Law – Formula

- ▶ The time to acquire a target is a function of the **distance** to and **size** of the target and depends on the particular pointing **system**

$$MT = a + b \log_2 \left(1 + \frac{D}{W} \right)$$

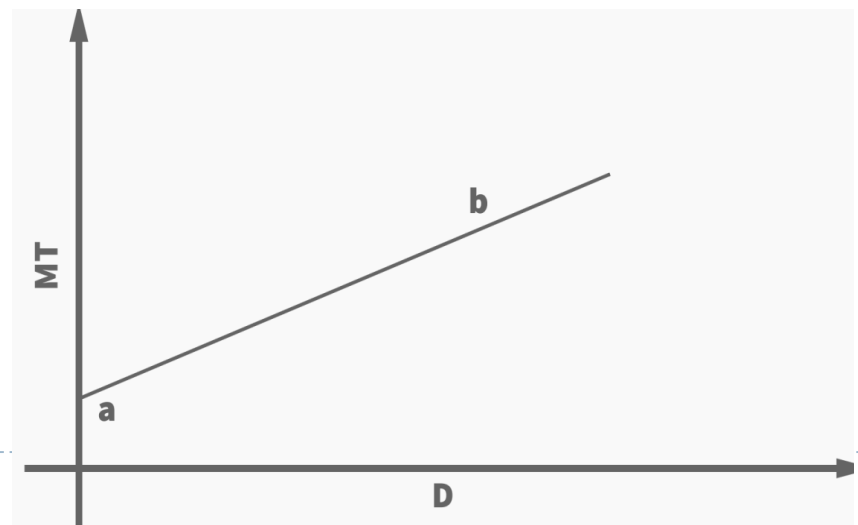


MT: movement time

a and **b**: constants dependent on the pointing system

D: distance to the target area

W: width of the target



▶ Index of Difficulty

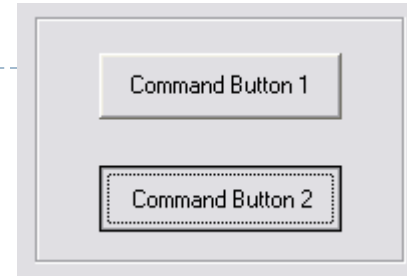
$$\text{Index of Difficulty, ID} = \log_2 \left(1 + \frac{D}{W} \right)$$

- ▶ $MT = a + b \cdot ID$
- ▶ ID describes the difficulty of the task
- ▶ independent of the device / method
- ▶ Index of Difficulty, ID measured in bits
- ▶ **Throughput - index of performance or bandwidth**
 - ▶ Single metric for input systems
 - ▶ One definition: $TP = ID / MT$ ('average' values of ID and MT are used)
 - ▶ Another definition: $TP = 1 / b$ (equals ID / MT only if $a=0$)

Design implications – Fitts' law

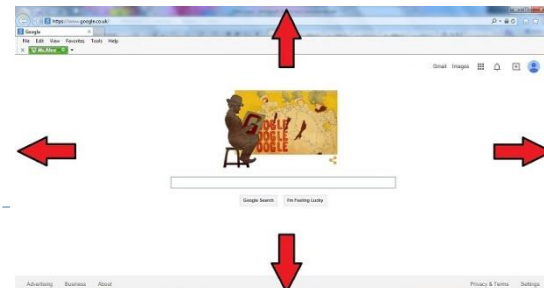
► Bigger buttons

- e.g. web links
- e.g. check / radio boxes
- any other interactive element in the graphical user interface must be distinguished from other non-interactive elements by size.



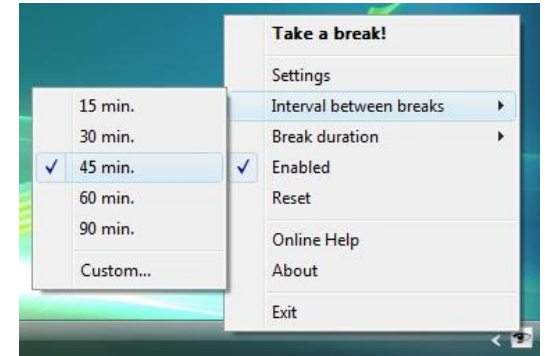
► Use edges and corners (for examples see next slide)

- edges of the screen have infinite height or width, respectively
- corners have infinite height and width.
- As the user is restricted in their movements the pointing device cannot move any further when they reach the outermost points of the screen



► Pop-up Menus

- These menus support immediate selection of interactive elements than dropdown menus as the user does not have to move the cursor from its current position.



► Muscular tension

- mobile devices are often carried around in pockets, which can trigger commands by accident. In those situations, high-precision input methods are deployed, which use a higher input precision to make sure that a command is not executed acci



References

- ▶ Nptel on human computer interaction
- ▶ Alan Dix, Human Computer Interaction: Practice

► Thank You