# Case Studies on Model-Based Design

Professor Ram Mohana Reddy Guddeti

Information Technology Department

NITK Surathkal, Mangalore, India

## Learning Objective

- In the previous lectures, we discussed the idea of models and model-based design
  - We discussed about different types of models used in HCI
  - We learned in details about four models, namely: KLM, (CMN)GOMS, Fitts’ law and Hick-Hyman law

## Learning Objective

- We have discussed the broad principles of model-based design
- In this and the following lecture, we shall see a specific case study on model-based design, namely design of virtual keyboards, to understand the idea better

## Virtual Keyboards

- Before going into the design, let us first try to understand virtual keyboard (VK)
- We know what a physical keyboard is
  - The input device through which you can input characters
- Although physical keyboards are ubiquitous and familiar, sometimes it is not available or feasible

# Virtual Keyboards

- Suppose you want to input characters in a  mobile device (e.g., your mobile phone or iPad)
  - Physical keyboards make the system bulky and reduces mobility
- Sometimes the users' may not have the requisite motor control to operate physical keyboards
  - For example, persons with cerebral palsy, paraplegia  etc.

## Virtual Keyboards

- In such scenario, VKs are useful
  - A VK is an on-screen representation of the physical keyboard (see the below image which shows text input in iPad with a VK)

## VK Design Challenge

- The iPad example in the previous slide show a QWERTY layout (i.e., key arrangement)
  - That's because the typing is two-hand and QWERTY layout is suitable for two-hand typing
- However, in many cases, VK is used with single-hand typing (particularly for small devices where one hand holds the device)

## VK Design Challenge

- Since the QWERTY layout is good for the two-hand typing, we have to find out the alternative "efficient" layout
  - Efficiency, in the context of keyboards in general and VK in particular, is measured in terms of character entry speed (characters/sec or CPS, words/min or WPM etc)

## VK Design Challenge

- Thus, what we want is a VK layout for single hand typing that allows the user to input characters with high speed and accuracy
- Mathematically, for a N character keyboard, we have to determine the best among N! possible key arrangements

## VK Design Challenge

- Thus, it is a typical “search” problem
  - We want to search for a solution in a search space of size N!
  - Note the “huge” size of the search space (for example, if N = 26 letters of English alphabet + 10 numerals = 36, the search space size is 36!)

## What We Can Do

- We can apply the standard design life cycle
- Drawbacks
  - We can not check all the alternatives in the search space (that will in fact take millions of years!)
- If the designer is experienced, he(she) can chose a small subset from the search space based on intuition

## What We Can Do

- The alternatives in the subset can be put through the standard design life cycle for comparison
  - However, empirical comparison still requires great time and effort
- Alternatively, we can use model-based approach to compare alternatives

## GOMS Analysis

- We can compare the designs in the subset using a GOMS analysis (also called CTA or cognitive task analysis)
- In order to do so, we first need to identify one or a set of "representative tasks"

## GOMS Analysis

- What is a task here?
  - To input a series (string) of characters with the VK
- Remember, we should have a representative task
  - That means, the string of characters that we chose should represent the language characteristics

## GOMS Analysis

- How to characterize a language?
- There are many ways
  - One simple approach is to consider unigram character distribution, which refers to the frequency of occurrence of characters in any arbitrary sample of the language (text)

## GOMS Analysis

- How to characterize a language?
- There are many ways
  - Bigram distribution, which refers to the frequency of occurrence of character pairs or bigrams in any arbitrary sample, is another popular way to characterize a language

## GOMS Analysis

- In order to perform GOMS analysis, we need to have character string(s) having language characteristics (say, the unigram distribution of characters in the string(s) match(es) to that of the language)
  - How to determine such string(s)?
- We can use a language corpus for the purpose

## Corpus

- Corpus (of a language) refers to a collection of texts sampled from different categories (genres)
  - Stories, prose, poem, technical articles, newspaper reports, mails …
- It is assumed that a corpus represents the language (by capturing its idiosyncrasies through proper sampling)

## Corpus

- However, corpus development is not trivial (requires great care to be truly representative)
- The good news is, already developed corpora are available for many languages (e.g., British National Corpus or BNC for English)
  - We can make use of those

## Corpus-based Approach

- How to use a corpus to extract representative text?
  - Get hold of a corpus
  - Extract a representative text through some statistical means (for example, cross-entropy based similarity measure)

# Cross-Entropy Based Similarity Measure

- Let X be a random variable which can take any character as its value
- Further, let P be the probability distribution function of X [i.e., $P(x_i) = P(X = x_i)$]
- We can calculate the "entropy", a statistical measure, of P in the following way

$$H(P) = -\sum_i P(x_i)\log_2 P(x_i)$$

# Cross-Entropy Based Similarity Measure

- Now, suppose there are two distributions, P and M
- We can calculate another statistical measure, called "cross-entropy", of the two distributions

$$H(P,M) = -\sum_i P(x_i)\log_2 M(x_i)$$

## Cross-Entropy Based Similarity Measure

- The cross-entropy measure can be used to determine similarity of the two distributions
  - Closer H(P,M) is to H(P), the better approximation M is of P (i.e., M is similar to P)
- We can use this idea to extract representative text from a corpus

## Cross-Entropy Based Similarity Measure

- Let P denotes the unigram probability distribution of the language
  - This can be determined from the corpus. Simply calculate the character frequencies in the corpus. Since the corpus is assumed to represent the language, the character frequencies obtained from the corpus can be taken as representative of the language
  - Calculate H(P)

## Cross-Entropy Based Similarity Measure

- Take random samples of texts from the corpus and determine the unigram character distribution of the sample text, which is M
- Next, calculate H(P,M)
- The sample text for which H(P,M) is closest to H(P) will be our representative text

## Problem with GOMS-based CTA

- Thus, we can perform GOMS analysis
- However, there is a problem
  - The text is usually large (typically >100 characters to make it *reasonably* representative), which makes it tedious to construct GOMS model

## Problem with GOMS-based CTA

- We need some other approach, which is not task-based, to address the design challenge
  - Task-based approaches are typically tedious and sometimes infeasible to perform
- In the next lecture, we shall discuss one such approach, which is based on the Fitts’ law and the Hick-Hyman law

## Learning Objective

- In the previous slides, we discussed the challenge faced by the Virtual Keyboards (VK) designers
  - The objective of the VK designer is to determine an efficient layout
  - The challenge for VK designer is to identify the layout from a large design space
  - We saw the difficulties in following the standard design life cycle
  - We explored the possibility of using GOMS in the design and discussed its problems

## Alternative Design Approach

- Here, Let us see another way of addressing the issue, which illustrates the power of model-based design
- We saw the problem with GOMS in VK design
  - The problem arises due to the task-based analysis, since identifying and analyzing tasks is tedious if not difficult and sometimes not feasible
- We need some other approach that is not task based
  - Fitts' Law and Hick-Hyman Law can be useful for the purpose as they do not require task-based analysis

## Fitts'-Digraph Model

- The alternative approach makes use of the Fitts' Diagraph (FD) model
- FD model was proposed to *compute* the user performance for a VK from layout specification
  - Layout in terms of keys and their positions
  - Performance in text entry rate

## Fitts'-Digraph Model

- The FD model has three components
  - **Visual Search Time (RT)**: time taken by a user to locate a key on the keyboard. The Hick-Hyman law is used to model this time

$$RT = a + b\log_2 N$$

N is the total number of keys, a and b are empirically-determined constants

# Fitts’-Digraph Model

- The FD model has three components
  - **Movement Time (MT)**: time taken by the user to move his hand/finger to the target key (from its current position). This time is modeled by the Fitts’ law

$$MT_{ij} = a' + b' \log_2 \left( \frac{d_{ij}}{w_j} + 1 \right)$$

MTij is the movement time from the source (i-th) to the target (j-th) key, dij is the distance between the source and target keys, wj is the width of the target key and a' and b' are empirically-determined constants

# Fitts'-Digraph Model

- The FD model has three components
  - **Digraph Probability:** probability of occurrence of character pairs or digraphs, which is determined from a corpus

$$P_{ij} = f_{ij} / \sum_{i=1}^{N}\sum_{j=1}^{N} f_{ij}$$

  - Pij is the probability of occurrence of the i-th and j-th key whereas fij is the frequency of the key pair in the corpus

# Fitts'-Digraph Model

- Using the movement time formulation between a pair of keys, an average (mean) movement time for the whole layout is computed

$$MT_{MEAN} = \sum_{i=1}^{N}\sum_{j=1}^{N} MT_{ij} \times P_{ij}$$

- The mean movement time is used, along with the visual search time, to compute user performance for the layout

## Fitts'-Digraph Model

- Users' Performance is measured in terms of the characters/second (CPS) or words/minute (WPM)
- Performances for two categories of users, namely: novice and expert users, are computed

## Fitts'-Digraph Model

- Novice User Performance: they are assumed to be unfamiliar with the layout. Hence, such users require time to search for the desired key before selecting the key

$$CPS_{Novice} = \frac{1}{RT + MT_{MEAN}}$$

$$WPM = CPS \times (60 / W_{AVG})$$

$W_{AVG}$ is the average number of characters in a word. For example, English words have 5 characters on average

# Fitts'-Digraph Model

- Expert User Performance: an expert user is assumed to be thoroughly familiar with the layout. Hence, such users don't require visual search time

$$CPS_{Expert} = \frac{1}{MT_{MEAN}}$$

$$WPM = CPS \times (60 / W_{AVG})$$

$W_{AVG}$ is the average number of characters in a word. For example, English words have 5 characters on average

## Using the FD Model

- If you are an expert designer
  - You have few designs in mind (experience and intuition helps)
  - Compute WPM for those
  - Compare

## Using the FD Model

- Otherwise
  - Perform *design space exploration* – search for a good design in the design space using algorithm
- Many algorithms are developed for design space exploration such as dynamic simulation, Metropolis algorithm and genetic algorithm
  - We shall discuss one (Metropolis algorithm) to illustrate the idea

## Metropolis Algorithm

- A “Monte Carlo” method widely used to search for the minimum energy (stable) state of molecules in statistical physics
- We map our problem (VK design) to a minimum-energy state finding problem in statistical physics

## Metropolis Algorithm

- We map a layout to a molecule (keys in the layout serves the role of atoms)
- We redefine performance as the average movement time, which is mapped to energy of the molecule
- Thus, our problem is to find a layout with minimum energy

## Metropolis Algorithm

- Steps of the algorithm
  - Random walk: pick a key and move in a random direction by a random amount to reach a new configuration (called a *state*)
  - Compute energy (average movement time) of the state
  - Decide whether to retain new state or not and iterate

## Metropolis Algorithm

- The decision to retain/ignore the new state is taken on the basis of the decision function, where ΔE indicates the energy difference between the new energy and old energy states
  (ΔE = energy of new state – energy of old state)

$$W(O-N)=\begin{cases} e^{-\frac{\Delta E}{kT}} & \Delta E>0 \\ 1 & \Delta E\le 0 \end{cases}$$

## Metropolis Algorithm

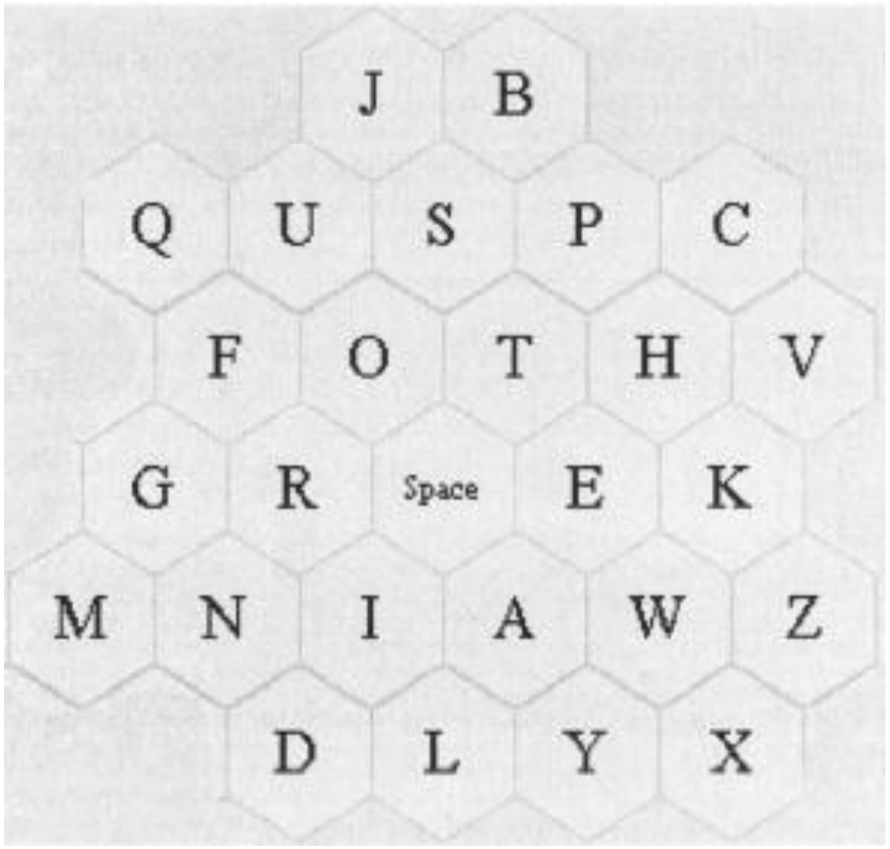
- W is probability of changing from old to new configuration
- k is a coefficient
- T is “temperature”
- Initial design: a “good” layout stretched over a “large” space

# Metropolis Algorithm

- Note the implications of the decision function
  - If energy of the new state is less than the current state, retain the new state
  - If the new state is having more energy than the current state, don’t discard the new state outright. Instead, retain the new state if the probability W is above some threshold value. This steps helps to avoid local minima
- To reduce the chances of getting struck at the local minima further, “Simulated Annealing” is used
  - Bringing “temperature” through several up & down cycles

# Metropolis Algorithm

An example VK layout, called the Metropolis layout, is shown, which was designed using the Metropolis algorithm

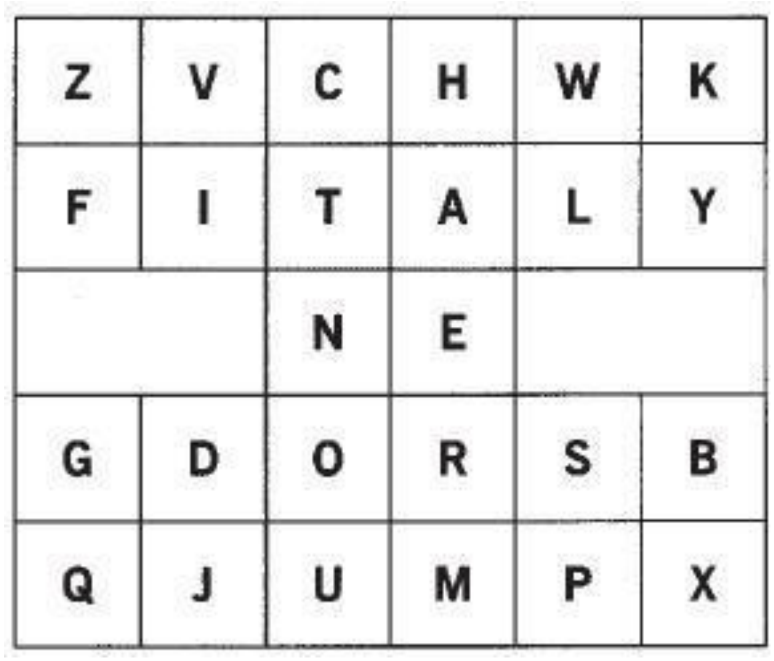


# Some VK Layouts with Performance

- QWERTY
  - 28 WPM (novice)
  - 45.7 WPM (expert)

- FITALY
  - 36 WPM (novice)
  - 58.8 WPM (expert)

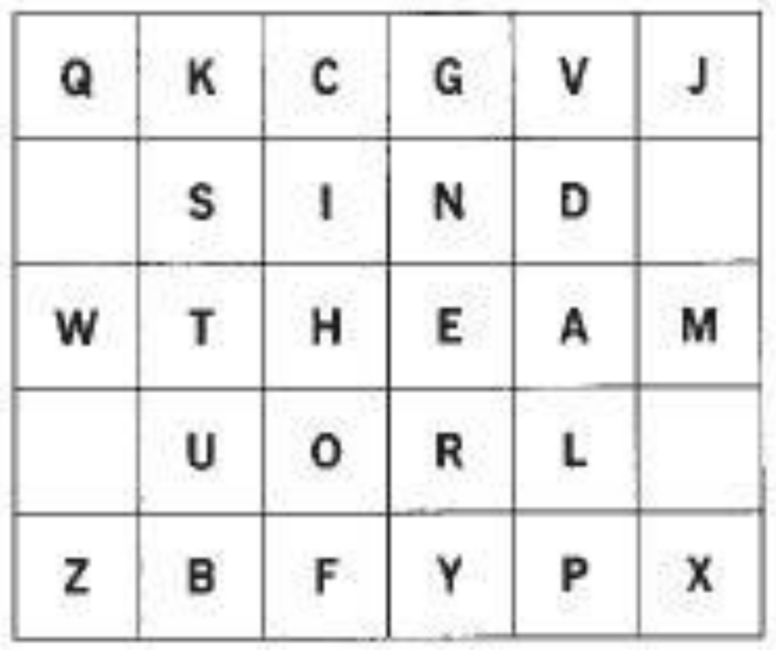
FITALY Keyboard



# Some VK Layouts with Performance

- QWERTY
  - 28 WPM (novice)
  - 45.7 WPM (expert)
- FITALY
  - 36 WPM (novice)
  - 58.8 WPM (expert)
- OPTI II
  - 38 WPM (novice)
  - 62 WPM (expert)

OPTI II Keyboard

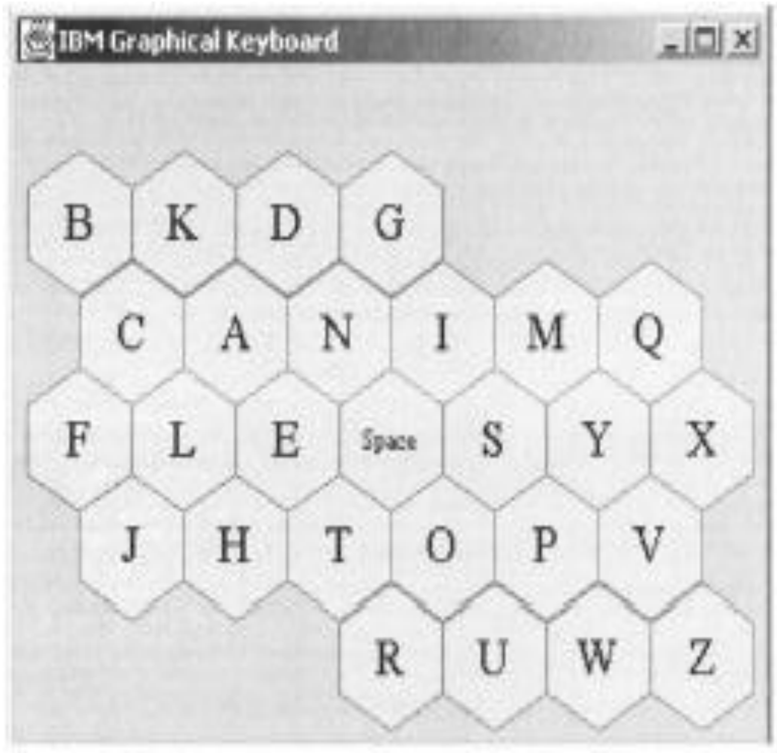


## Some VK Layouts with Performance

- The layouts mentioned before were not designed using models
- They were designed primarily based on designer’s intuition and empirical studies
- However, the performances shown are computed using the FD model

# Some VK Layouts with Performance

- ATOMIK – a layout designed using slightly modified Metropolis algorithm
- Performance of the ATOMIK layout
  - 41.2 WPM (novice)
  - 67.2 WPM (expert)



## Some VK Layouts with Performance

- Note the large performance difference between the ATOMIK and other layouts
- This shows the power of model-based design, namely a (significant) improvement in performance without increasing design time and effort (since the design can be mostly automated)