

# Introduction to HCI

## Fall 2021

# Evaluation

## Heuristic Evaluation

Mahmood Jasim

UMass Amherst

[mjasim@umass.edu](mailto:mjasim@umass.edu)

<https://people.cs.umass.edu/~mjasim/>

© Mahyar with acknowledgements to Joanna McGrenere and Dongwook Yoon

# Logistics

- ▶ Milestone 2 report due tomorrow
- ▶ Milestone 3 will be posted tomorrow
- ▶ Midterm grading is close to finish
  - ▶ Will be disseminated next Tuesday
- ▶ Time to code!

# Learning goals

- ▶ Know how to apply heuristics
- ▶ Explain Heuristic evaluation
- ▶ Describe the pros/cons of heuristic evaluation

# Heuristic evaluation

## ▶ What for:

- ▶ Identifying (listing & describing) problems with existing prototypes (any kind of interface); for any kind of user, new or proficient

## ▶ Cost-benefit:

- ▶ Usability engineering activities are often expensive / slow; but some can be quick / cheap, and still produce useful results
- ▶ Focused less on what is "correct" than on what can be done within development constraints
- ▶ Ultimate trade-off may be between doing no usability assessment and doing some kind

# How to preform a heuristic evaluation

- ▶ Design team supplies scenarios, prototype, list of heuristics
- ▶ Need 3-5 evaluators: train in method if non-expert
  - ▶ Single evaluator catches ~35% of the usability problems
  - ▶ Five evaluators catch ~75%
- ▶ Each evaluator independently produces list of justified, rated problems by stepping through interface and applying heuristics at each point
  - ▶ use heuristics list & severity rating convention
- ▶ Team meets and compiles report that organizes and categorizes problems

# Individuals vs. teams

- ▶ Nielsen recommends individual evaluators inspect the interface alone.
  - ▶ evaluation is not influenced by others
  - ▶ independent and unbiased
  - ▶ greater variability in the kinds of errors found
  - ▶ no overhead required to organize group meetings

# Step 1: briefing session

- ▶ Get your experts together
  - ▶ Brief them on what to do, goals of system, etc.
  - ▶ Discuss heuristics to be applied
- ▶ May also want to provide experts with:
  - ▶ Some examples of tasks
  - ▶ Descriptions of user personas
  - ▶ Simple instructions/guidance
    - ▶ Especially if NOT a fully functioning system

## Step 2: individual evaluation

- ▶ At least two passes for each evaluator
  - ▶ First to get feel for flow and scope of system
  - ▶ Second to focus on specific elements
- ▶ Each evaluator produces list of problems
  - ▶ Explain problem w/reference to heuristic or other info
  - ▶ Be specific and list each problem separately
  - ▶ Assign rating of **severity** to each violation



# Evaluation form

**Example Heuristic Evaluation Form**

Evaluator: \_\_\_\_\_ Prototype: \_\_\_\_\_ Date/Time: \_\_\_\_\_ Pg: \_\_\_ / \_\_\_

Heuristic violated	Description / Comment	Severity

1

# Severity ratings

- ▶ Each violation is assigned a **severity rating**
  - ▶ Many other methods of doing this
- ▶ Usually some combination of:
  - ▶ Frequency
  - ▶ Impact
  - ▶ Persistence (one time or repeating)
- ▶ Used to:
  - ▶ Help prioritize problems
  - ▶ Allocate resources to fix problems
  - ▶ Estimate need for more usability efforts
- ▶ Can be done independently by all evaluators or later as group prioritizes

# Example severity & extent scales

## one severity scale:

- ▶ 0 - don't agree that this is a usability problem
- ▶ 1 - cosmetic problem
- ▶ 2 - minor usability problem
- ▶ 3 - major usability problem; important to fix
- ▶ 4 - usability catastrophe; imperative to fix

## one extent scale:

- ▶ 1 = single case
- ▶ 2 = several places
- ▶ 3 = widespread

## Step 3: aggregating results & making recommendations

- ▶ Evaluation team meets and compares results
- ▶ Through discussion and consensus, each violation is documented and categorized in terms of severity, extent
- ▶ Violations are ordered in terms of severity
  - ▶ E.g., Use an excel spreadsheet (which can be sorted)
- ▶ Combined report goes back to design team

# Heuristic evaluation

## Advantages

- ▶ Contributes valuable insights from objective observers
- ▶ The “minimalist” approach
  - ▶ General guidelines can correct for majority of usability problems
  - ▶ Easily remembered, easily applied with modest effort
  - ▶ Systematic technique that is reproducible with care
- ▶ Discount usability engineering
  - ▶ Cheap and fast way to inspect a system
  - ▶ Can be done by usability experts and rapidly-trained end users

# Heuristic evaluation

## Problems:

- ▶ Principles must be applied intuitively and carefully
- ▶ Can't be treated as a simple checklist
- ▶ Heuristics can narrow focus on some problems at cost of others
- ▶ Can reinforce existing design (not for coming up with radical ideas)
- ▶ Doesn't necessarily predict users/customers' overall satisfaction
- ▶ May not have same "credibility" as user test data

# Combining Heuristic Evaluation and Cognitive Walkthrough

- ▶ HCI practitioners often use a combination of both that might vary based on what they're trying to learn
  - ▶ e.g., While doing a walkthrough for a task, apply the heuristics at each step, in addition to the CW questions.

# One popular list of heuristics (Nielsen, '93)

- ▶ H1: Visibility of system status
- ▶ H2: Match between system & the real world
- ▶ H3: User control & freedom
- ▶ H4: Consistency and standards
- ▶ H5: Recognition rather than recall
- ▶ H6: Error prevention
- ▶ H7: Flexibility and efficiency of use
- ▶ H8: Aesthetic and minimalist design
- ▶ H9: Help users recognize, diagnose & recover
- ▶ H10: Help and documentation



1



Visibility of system status

2



Match between system + real world

3



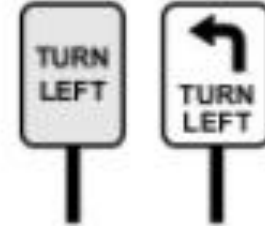
User control and freedom

4



Consistency and standards

5



Recognition rather than recall

6



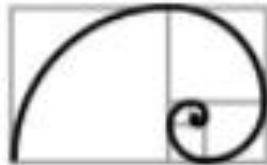
Error prevention

7



Flexibility and efficiency of use

8



Aesthetic and minimalist design

9



Help users with errors

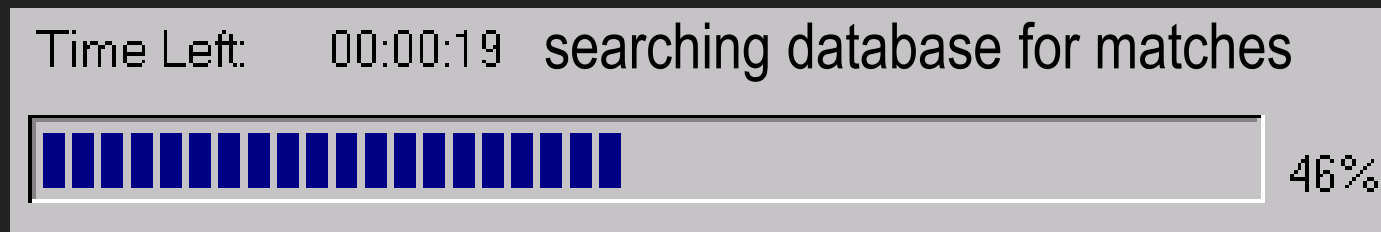
10



Help and documentation

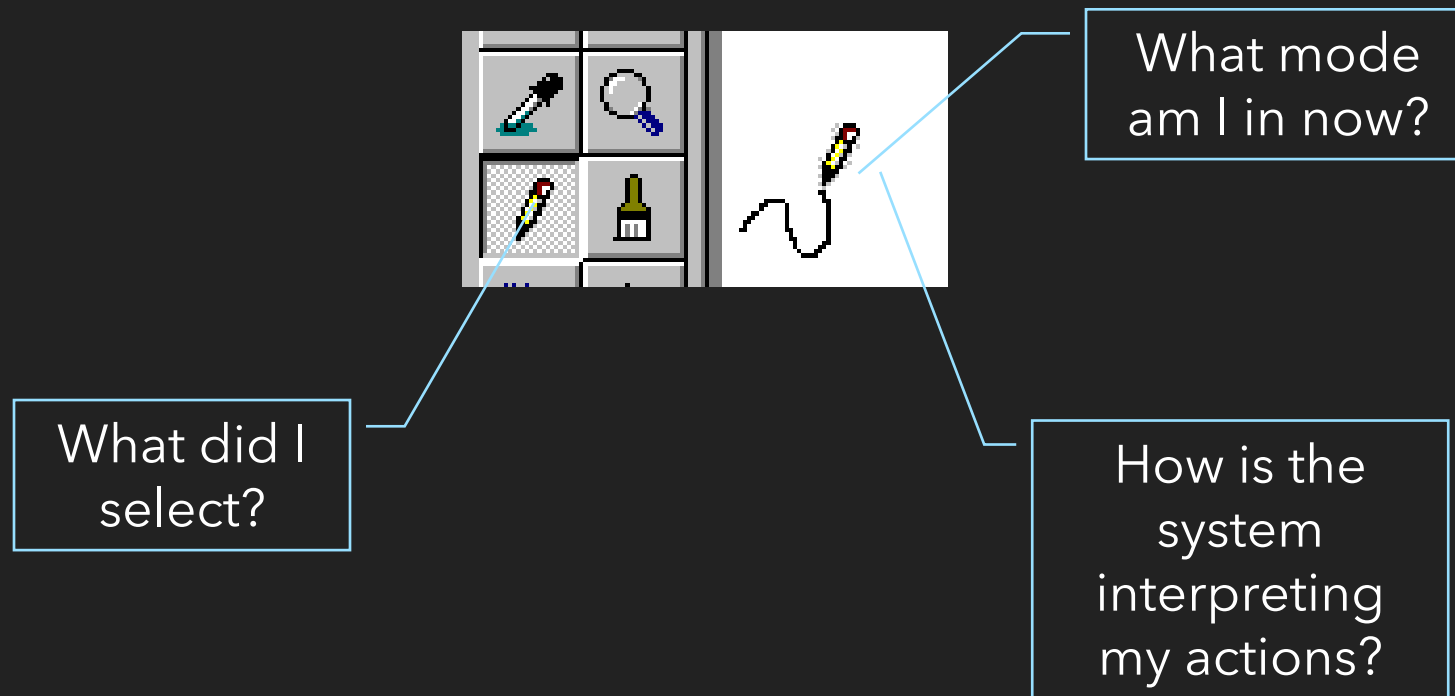
# H1: Visibility of system status

- ▶ The system should always keep users informed about what is going on, through (appropriate feedback within reasonable time)
- ▶ Example: consider system response time (user must wait)
  - ▶ 0.1 sec: no special indicators needed, why?
  - ▶ 1.0 sec: user starts to lose track of data, objects, etc.
  - ▶ 10 sec: max duration if user to stay focused on action
  - ▶ for longer delays, use percent-done progress bars



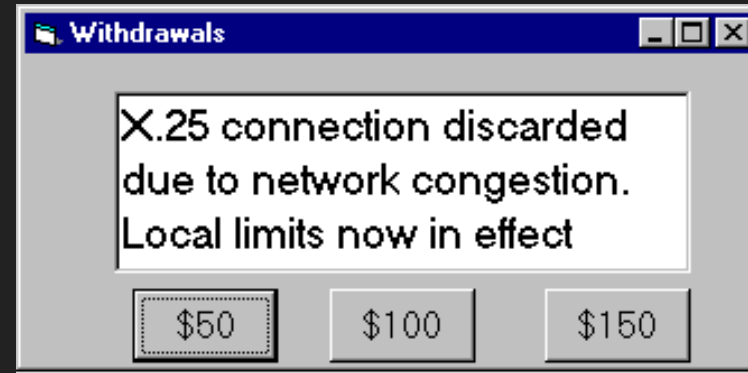
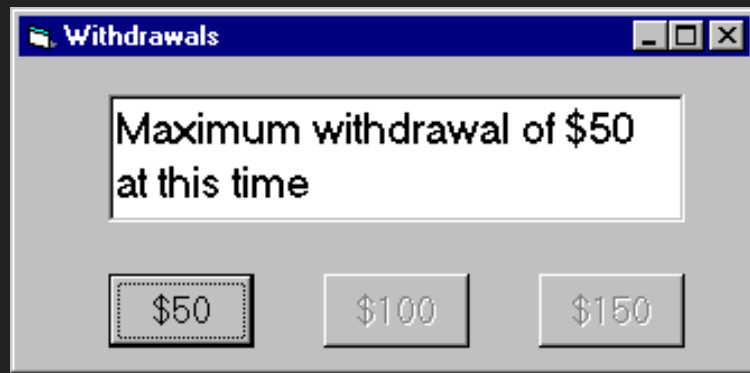
# H1: Visibility of system status

- ▶ Keep users informed about what is going on
  - ▶ Appropriate visible feedback

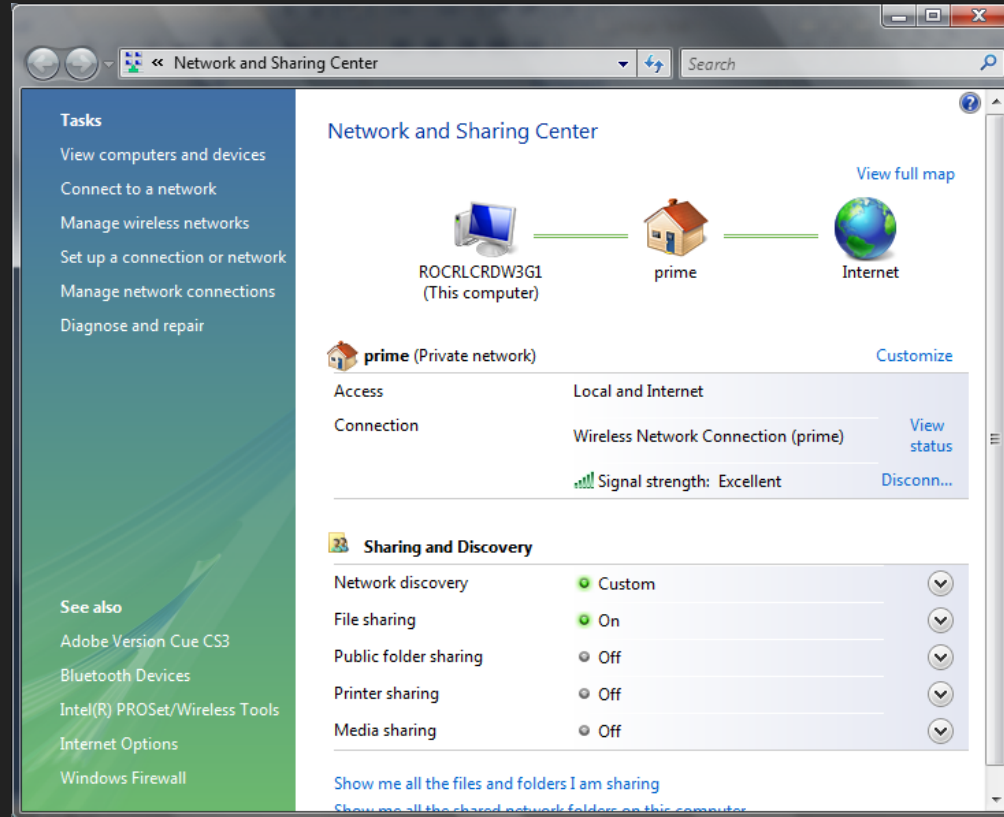
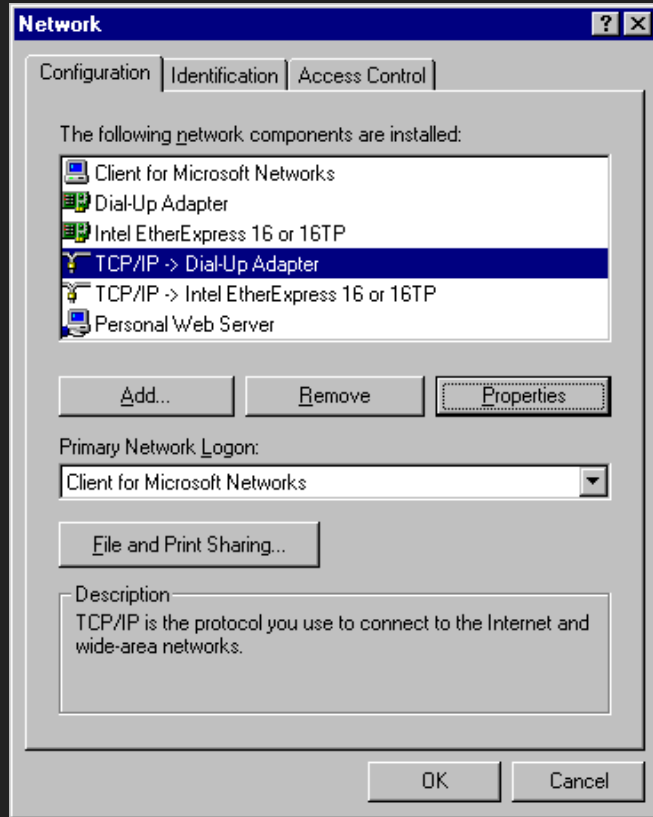


## H2: Match between system & real world

- ▶ The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
  - ▶ e.g. withdrawing money from a bank machine



# H2: Match between system & real world



## H3: User control & freedom

Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

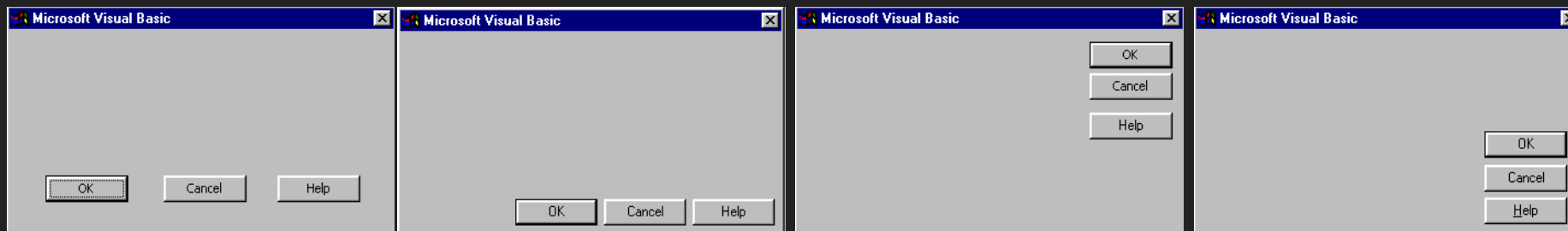


# H3: User control & freedom

- ▶ “Exits” for mistaken choices, undo, redo
- ▶ Don’t force down fixed paths
  
- ▶ Strategies:
  - ▶ Cancel button (for dialogs waiting for user input)
  - ▶ Universal undo (can get back to previous state)
  - ▶ Interrupt (especially for lengthy operations)
  - ▶ Quit (for leaving the program at any time)
  - ▶ Defaults (for restoring a property sheet)

# H4: Consistency & standards

- ▶ Consistency of effects = **predictability**
  - ▶ Same words, commands, actions should always have the same effect in equivalent situations
- ▶ Consistency of language and graphics
  - ▶ Same info/controls in same location on all screens/dialog boxes:

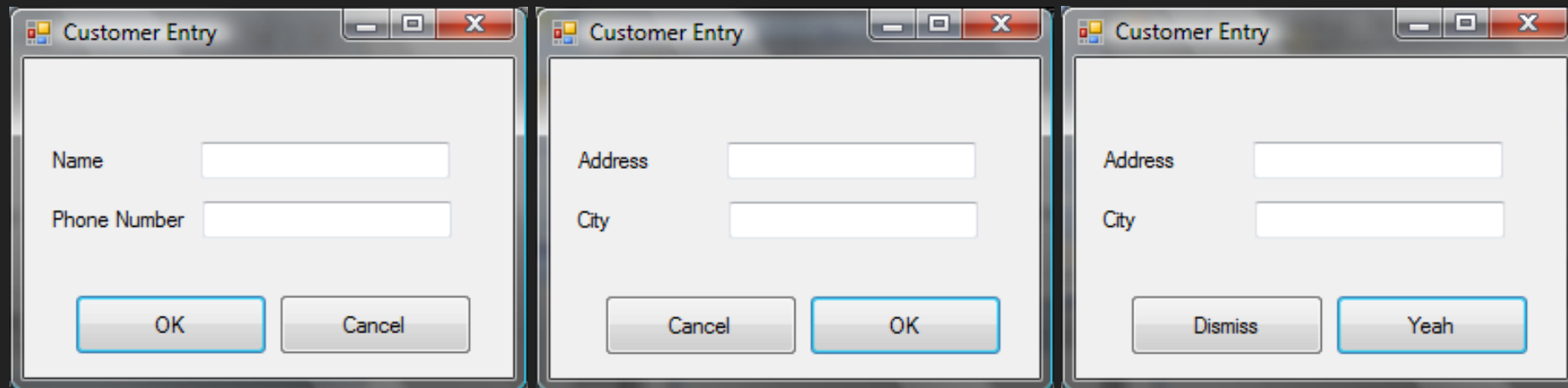


- ▶ Same visual appearance across the system (e.g., widgets)
    - ▶ e.g., not different scroll bars in a single window system
- ▶ Consistency of input
  - ▶ require consistent syntax across complete system



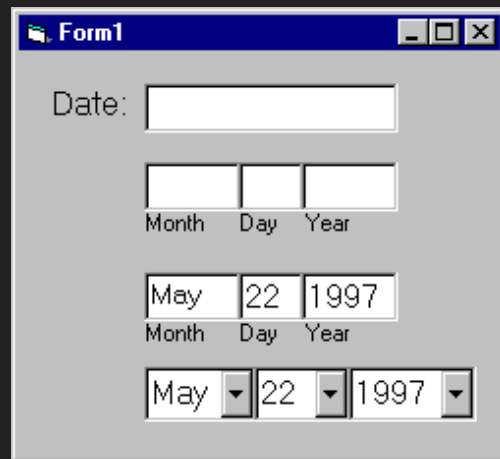
# H4: Consistency & standards

- ▶ Consistency of language and graphics
  - ▶ Same info/controls in same location on all screens/dialog boxes

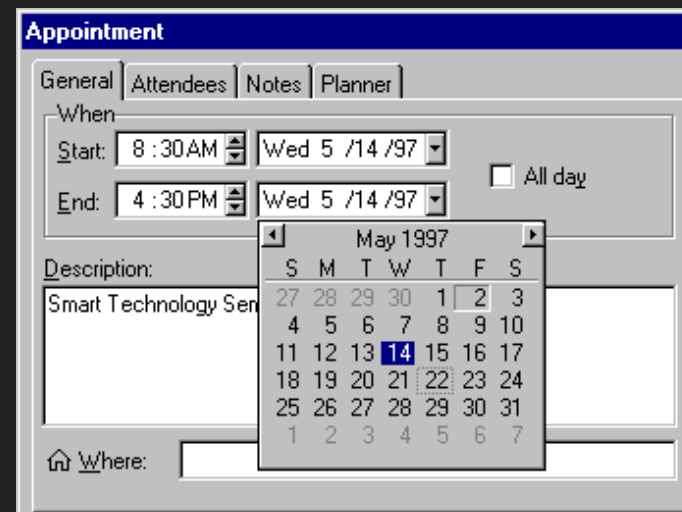


# H5: Error prevention

- ▶ Try to make errors impossible
  - ▶ Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
- ▶ Modern widgets: only “legal commands” selected, or “legal data” entered



A screenshot of a window titled "Form1" showing four different date input widgets. The first is a single text box labeled "Date:". The second consists of three separate boxes for "Month", "Day", and "Year". The third is a text box containing "May 22 1997" with "Month", "Day", and "Year" labels below it. The fourth is a dropdown menu with "May", "22", and "1997" selected, with "Month", "Day", and "Year" labels below it.



A screenshot of an "Appointment" form with tabs for "General", "Attendees", "Notes", and "Planner". The "When" section has "Start" and "End" fields with time and date pickers. A calendar widget for "May 1997" is open, showing the date "14" selected. The "Description" field contains "Smart Technology Ser". A "Where:" field is at the bottom.

May 1997						
S	M	T	W	T	F	S
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

# H5: Errors we make

## ▶ Mistakes

- ▶ Arise from conscious deliberations that lead to an error instead of the correct solution

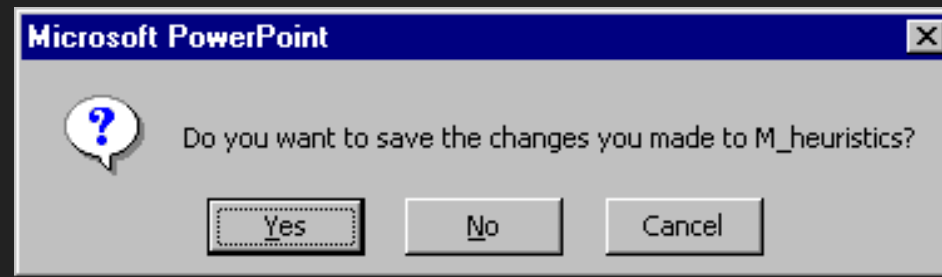
## ▶ Slips

- ▶ Unconscious behavior that gets misdirected en route to satisfying goal
  - ▶ e.g., Drive to store, end up in the office
- ▶ Shows up frequently in skilled behavior
  - ▶ Usually due to inattention
- ▶ Often arises from similarities of actions

# H5: Types of slips

## ▶ Capture error

- ▶ Frequent response overrides [unusual] intended one
- ▶ Occurs when both actions have the same initial sequence
  - ▶ Confirm saving of a file when you don't want to delete old version



# H5: Types of slips

## ▶ Description error

- ▶ Intended action has too much in common with others possible
- ▶ E.G. When right and wrong objects physically near each other
  - ▶ Pour juice into bowl instead of glass
  - ▶ Go jogging, come home, throw sweaty shirt in toilet instead of laundry
  - ▶ Move file to trash instead of to folder

## ▶ Loss of activation

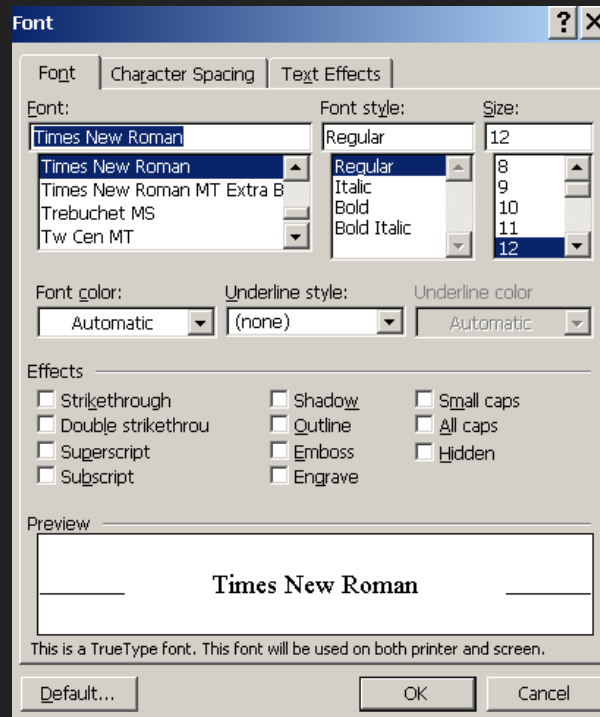
- ▶ Forgetting the goal while carrying out the action sequence
- ▶ e.g., Start going to a room and forget why by the time you get there
  - ▶ Navigating menus/dialogs, can't remember what you are looking for
  - ▶ But continue action to remember (or go back to beginning)!

## ▶ Mode errors

- ▶ People do actions in one mode thinking they are in another
  - ▶ Refer to file that's in a different directory
  - ▶ Look for commands / menu options that are not relevant

# H6: recognition rather than recall

- ▶ Computers are good at remembering things, people aren't!
- ▶ Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.



# H7: flexibility and efficiency of use

- ▶ Experienced users should be able to perform frequently used operations quickly
- ▶ **Strategies:**
  - ▶ Keyboard and mouse accelerators
    - ▶ Abbreviations
    - ▶ Command completion
    - ▶ Menu shortcuts & function keys
    - ▶ Double clicking vs. Menu selection
  - ▶ Type-ahead (entering input before the system is ready for it)
  - ▶ Navigation jumps
  - ▶ Go to desired location directly, avoiding intermediate nodes
  - ▶ History systems
    - ▶ WWW: ~60% of pages are revisits

# H8: aesthetic and minimalist design

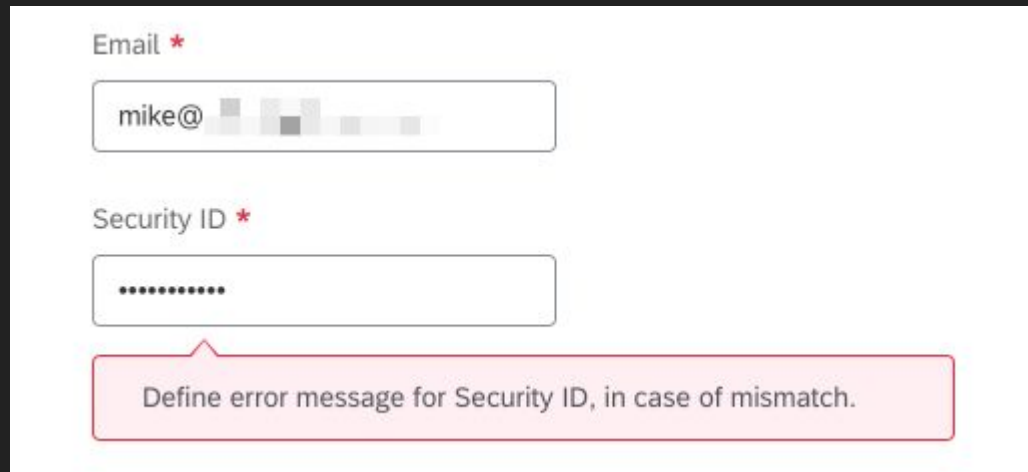
- ▶ Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Form Title -- (appears above URL in most browsers and is used by WWW search)		Background Color:
Q&D Software Development Order Desk		FFFBF0
Form Heading -- (appears at top of Web page in bold type)		Text Color:
Q&D Software Development Order Desk <input checked="" type="checkbox"/> Center		000080
E-Mail responses to (will not appear on)	Alternate (for mailto forms only)	Background Graphic
dversch@q-d.com		
Text to appear in Submit button	Text to appear in Reset button	<input type="radio"/> Mailto
Send Order	Clear Form	<input checked="" type="radio"/> CGI
Scrolling Status Bar Message (max length = 200 characters)		
****WebMania 1.5b with Image Map Wizard is here!****		
<input type="button" value=" &lt;&lt; Prev Tab"/>		<input type="button" value=" Next Tab &gt;&gt;"/>



## H9: help users recognize, diagnose, and recover from errors

- ▶ Error messages should be expressed in plain language (no codes), precisely
- ▶ Indicate the problem, and constructively suggest a solution.



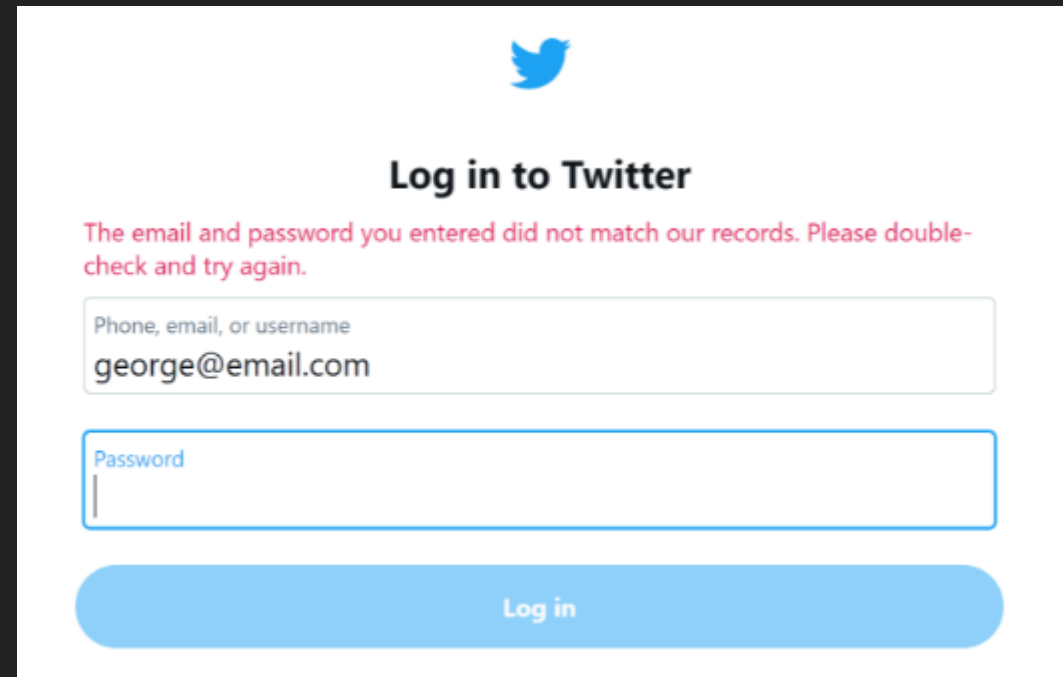
Email \*


  
  

Security ID \*

Define error message for Security ID, in case of mismatch.





### Log in to Twitter

The email and password you entered did not match our records. Please double-check and try again.

Phone, email, or username  
george@email.com

Password

Log in

# H10: help and documentation

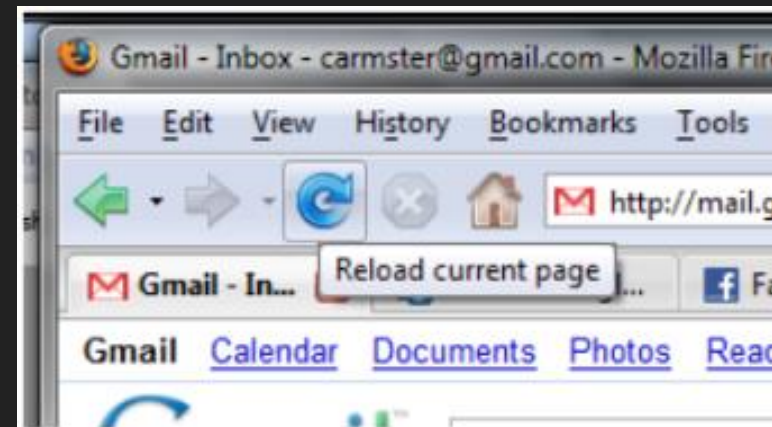
- ▶ Help is not a replacement for bad design!
- ▶ Simple systems: walk up and use; minimal instructions
- ▶ Most other systems:
  - ▶ Feature-rich
  - ▶ Some users want to become “expert” rather than “casual” users
  - ▶ Intermediate users need reminding, plus a learning path
- ▶ Many users do not read manuals
- ▶ Usually used when users are panicked & need help NOW
  - ▶ Need online documentation, good search/lookup tools
  - ▶ Online help can be specific to current context
- ▶ Sometimes used for quick reference
  - ▶ Syntax of actions, possibilities
  - ▶ List of shortcuts

# H10: types of help

- ▶ Tutorial and/or getting started manuals
  - ▶ Short guides that people usually read when first encounter system
    - ▶ Encourage exploration and getting to know the system
    - ▶ Communicate conceptual material and essential syntax
  - ▶ On-line "tours", exercises, and demos
    - ▶ Demonstrate very basic principles through working examples
- ▶ Reference manuals
  - ▶ Used mostly for detailed lookup by experts
  - ▶ Rarely introduces concepts
    - ▶ Thematically arranged
  - ▶ On-line hypertext
    - ▶ Search / find
    - ▶ Table of contents
    - ▶ Index
    - ▶ Cross-index

# H10: types of help (cont'd)

- ▶ Reminders
- ▶ Short reference cards
  - ▶ Expert user who just wants to check facts
  - ▶ Novice who wants to get overview of system's capabilities
- ▶ Keyboard templates
  - ▶ Shortcuts/syntactic meanings of keys; recognition vs. Recall; capabilities
- ▶ Tooltips
  - ▶ Text over graphical items indicates their meaning or purpose



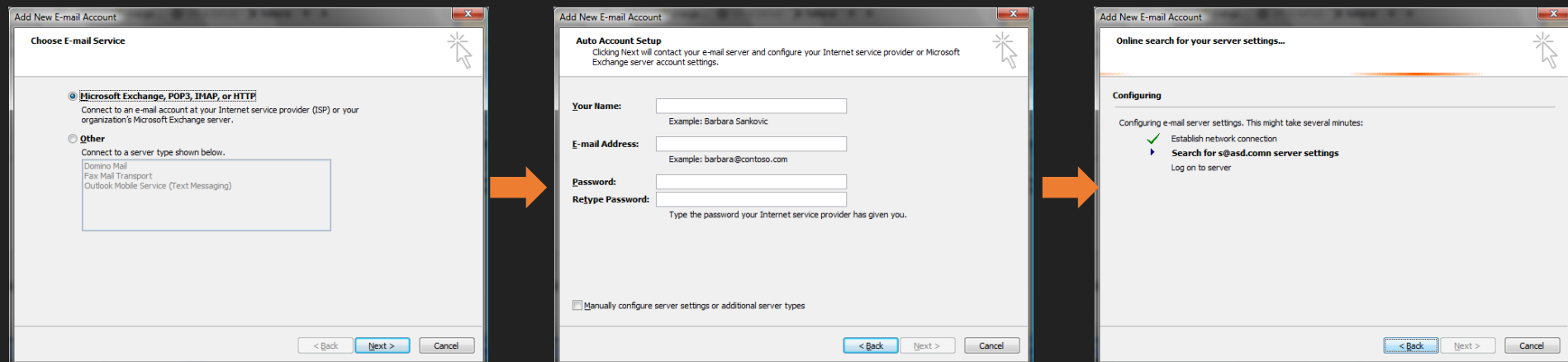
# H10: types of help

## ▶ Context-sensitive help

- ▶ System provides help on the interface component the user is currently working with
  - ▶ Macintosh "balloon help"
  - ▶ Microsoft "what's this" help

## ▶ Wizards

- ▶ Walks user through typical tasks
- ▶ Reduces user autonomy



# In-class activity

- ▶ An example system, CommunityPulse - (<https://communitypulse.cs.umass.edu>):
- ▶ A visual analytic system that utilizes text analysis to extract important topics, emotions and sentiments from community comments and enables civic leaders to explore the comments at multiple levels of granularity

# In-class activity

- ▶ Work in groups
- ▶ Write down participating members' names
- ▶ Perform heuristic evaluation based on the 10 heuristics
- ▶ Link to worksheet: <https://tinyurl.com/jz7fh3b4>

# Optional Reading

- ▶ Ten Usability Heuristics for User Interface Design
  - ▶ <https://www.nngroup.com/articles/ten-usability-heuristics/>
- ▶ Heuristic Evaluation
  - ▶ <https://drive.google.com/file/d/1LFOZhjoufAcAXkBhYF3t7gUIDv8yZniN/view?usp=sharing>