Questions?

• Project #3 due this Thursday
• HW #7 due April 24
Qualitative approach

- Gather users perception of the interaction

Methods
- Without users
  - Walkthroughs
  - Nielsen's heuristics based evaluation
- Direct observation
  - Simple observation
  - Thinking aloud
  - Constructive interaction (co-discovery)
- Interviews, questionnaires and surveys
Walkthrough

• Designer tries the system (or prototype) out
  – Does the system “feel right”? 
  – What if?

• Pros and Cons
  – Very simple to do
  – Completely subjective
  – Designer is a non-typical user

From “The inmates are running the Asylum” (A Cooper)
Nielsen's heuristics based evaluation

• “Rules of thumb” that describe features of usable systems
  – Can be used as design principles
  – Can be used to evaluate a design

• Pros and cons
  – Easy and inexpensive
    • Performed by expert
    • No users required
    • Catch many design flaws
  – More difficult than it seems
    • Not a simple checklist
    • Cannot assess how well the interface will address user goals
Usability Engineering

• Introduced by Nielsen (1994)
• Can be performed on working UI or sketches
• Required a small set (3-5) of evaluators to examine the UI
  – Check compliance with usability principles
    • Each evaluator works independently
    • Go through the interface several times
  – All reviews are aggregated in one final usability report
Nielsen's evaluation phases (1-2)

• Pre-evaluation training
  – Provide the evaluator with domain knowledge if needed

• Evaluation
  – First step: get a feel for flow and scope
  – Second step: focus on specific elements
    • Multiple passes approach is better
    • Create a list of all problems
Nielsen's evaluation phases (3-4)

• Severity rating
  – Performed after individual evaluations are aggregated
  – Establishes a ranking between problem
  – Reflects frequency, impact and persistence
    • Cosmetic, minor, major and catastrophic

• Debriefing
  – Discuss outcome with design team
  – Suggest potential solutions
  – Assess how hard things are to fix
Neilsen’s heuristics

- Simple and natural dialog
- Speak the users’ language
- Minimize user memory load
- Consistency
- Feedback
- Clearly marked exits
- Shortcuts
- Prevent errors
- Good error messages
- Provide help and documentation
Simple and natural dialog

• Present information in natural order

• Occam’s razor
  – Remove or hide irrelevant or rarely needed information
    • They compete with important information on screen
  – Use windows frugally
    • Avoid complex window management

From Cooper’s “About face 2.0”
Speak the users’ language

• Use a language compatible with users’ conceptual model
  – Example: withdrawing money at an ATM

• Use meaningful mnemonics, icons and abbreviations
Minimize user memory load

• Promote recognition over recall
  – Recognition is easier than recall

• Describe expected input clearly
  – Don’t allow for incorrect input
Consistency

• Be consistent in
  – Command design
    • Same action, same effect in equivalent situations
  – Graphic design
    • Input format
    • Output format
  – Flow design
    • Similar tasks are handled in similar ways

• Consistency promotes skills acquisition and/or transfer
Feedback (Semantic)

• Users should always be aware of what is going on
  – So that they can make informed decision
    • Be specific
  – But do not overburden users!
  – Provide redundant information

Feedback: Toolbar, cursor, ink
Feedback (Time)

• Different feedback time scales
  – Shall I wait for that task to finish or go for coffee?
    
    .1s  Causality
    1s  Delay but user’s flow of thought is uninterrupted
    10s  Difficult to stay focused
    > 10s  User will switch to another task while waiting

• Different techniques
  – Short transaction: hour glass cursor
  – Longer transaction: estimate of time left
    • An overestimate is always better!
Clearly marked exits

• Users don’t like to be trapped!

• Strategies
  – Cancel button (or Esc key) for dialog
    • *Make the cancel button responsive!*
  – Universal undo
Shortcuts

• Expert users should be able to perform operations rapidly
  – Try to limit the training necessary to access advanced features

• Strategies
  – Keyboard and mouse accelerators
    • menu shortcuts and function keys
    • command completion, command abbreviations and type-ahead
  – Toolbars and tool palettes
    • Trade screen real estate for rapid access
  – Navigation jumps
    • History systems
      – 60% pages are revisits
Preventing errors

• Error types
  – Mistakes
    • Conscious decision with unforeseen consequences

  – Slips
    • Automatic behaviors kicking in
      – Drive to the store, end-up in the office
      – Press enter one time too many…
    • Mode errors
      – Forget the mode the application is in
    • Loss of activation
      – Forget what your goals were
Forcing functions

One once of prevention is worth more than a pound of cure!

- **Interlock mechanisms**
  - Switching from P to D in a car

- **Lockin mechanisms**
  - No eject button for floppy disk on Mac

- **Lockout mechanisms**
  - Exit stairways
Designing for slips

• Avoid mode as much as possible

• Prevent users from entering dangerous state

• Check for reasonable input
  – Be prepared to handle several formats
  – Make entering a incorrect format impossible

• Make the current goal clear
  – Prevent lost of activations
Dealing with errors

• People will make errors!
  – You can ignored them
    • Generally very confusing
  – You can correct them automatically
    • Spelling corrector
    • But will I trust the system to be right 100%
  – You can discuss about it
    • But novice/expert tradeoff
  – You can try to teach the user what to do
    • Office assistant

• Respect users feelings!
Good error messages

• Provide meaningful error messages
  – Explain the problem in terms of or user conceptual model
  – Don’t make the user feel stupid
  – Offer a way to correct the problem

  – Compare
    • Error 25: access denied
    • Cannot open “chapter 5” because “Microsoft Word” is not installed. Do you want to use Notepad instead?
Good error messages

From Cooper’s “About Face 2.0”
Provide help and documentation

• Providing help is not an excuse for poor design!
  – Saving a couple of line of code or writing several pages of documentation?
  – Users don’t like to read manuals
    • *They prefer to learn while making progress toward their goals*

• Most users will stay at the intermediate level
  – Need reminders and a clear learning path
  – Need a quick way to access critical information
    • *Online documentation and good search tool*