Lecture 12: Beyond WIMP

WIMP interfaces

- WIMP = Windows, Icons, Menus, Pointers
  - they're everywhere!
  - in fact, after circa 1983, the vast majority of interfaces are based on the WIMP paradigm
    - e.g., think about Windows vs. Macintosh vs. UNIX-based window managers
  - some exceptions...
    - e.g., text terminals, game systems
  - ... but generally, WIMPs rule!

Beyond WIMP

- WIMP has been around a while, and probably will remain on the scene for a long time.
- But let’s think forward. What’s the next step?
  - Nielsen: “virtual realities, head-mounted displays, sound and speech, pen and gesture recognition, animation and multimedia, limited artificial intelligence, and highly portable computers with cellular or other wireless communication capabilities” (?!)
  - increasing computing power makes this possible
  - but can all this be in a single interface? maybe not... that’s yet another difference!
  - interfaces may become more task/user specific
Functional vs. Object-Oriented

• "Old-school" interfaces are functional in requiring specification of entire function
  - uses a "verb-noun" syntax
  - e.g., "rm foo", "emacs file.java"
• Current GUIs are object-oriented
  - uses a "noun-verb" syntax
  - e.g., select icon, drag to trash / select "Open"
• What seems to be coming on the horizon?
  - "syntax-free" interfaces
  - well, at least syntax-flexible
    ▪ can specify noun-verb, or verb-noun, or something radically different... just like communication w/ people

One step forward...

• Thought exercise
  - imagine you have access to a movie database
    ▪ database = info about films, actor/actresses, etc.
    ▪ how can you answer questions like...
      ▪ In what films did Harrison Ford star between 1980-90?
      ▪ In what films did Julia Roberts and Richard Gere costar?
      ▪ What are the most popular Sci Fi movies of 1993?

One step forward...

• Ahlberg & Shneiderman’s FilmFinder

Non-command interfaces

• Idea: user and computer interact not through a dialogue of commands and responses, but through a complex, dynamic, continuous interaction
• Huh? What’s that?
• Nielsen’s 12 “interaction characteristics” for next-generation, non-command software
  - not all applications will incorporate all 12
  - rather, it is expected that many applications will incorporate a significant subset of the 12
1. User focus
• Interaction feels like “using a computer”, not working on a task (according to Nielsen)
• With NC interfaces, focus = task… features come for free implicitly
• Example: Portholes system
  - update every 5min
  - implicit awareness

2. Computer’s role
• Old: “Do what you’re told”
  New: “Do what I mean!”
• Great idea… but why is this hard?
• Example: Intelligent tutoring
  - monitor what the student knows
  - interrupt with instruction when necessary
• Example: Model tracing / “Mind tracking”
  - infer student knowledge, or disabled user’s intentions, or driver’s intentions...

3. Interface control
• Old: User controls computer
  New: Computer controls
• Examples: warn user of incoming email, infer current writing task and provide template, etc.
• BUT this is very hard to do well
  - must avoid interrupting the user
  - guesses / inferences had better be right!

4. Syntax
• Old: Rigid interaction “syntax”
  New: No / little syntax
• Example: deleting files
  - way #1: select and delete (noun-verb)
  - way #2: say “remove all *.java files” (verb-noun)
  - can we integrate multiple methods?
• Example: writing math expressions
  - try not to require top-down or bottom-up
  \[ \sqrt{\frac{x + y}{2}} \]
5. Object visibility

- Old-school interfaces with “direct manipulation” require visible objects
- New interfaces could manipulate objects implicitly through higher-level interactions, or with hidden agents
- Might this be dangerous?
  - user doesn’t know about manipulation
  - can be good, can be bad

6. Interaction stream

- Old: Single-threaded input / output
  - New: Multi-threaded, multimodal
- Example: “Put that there!”
  - point to display wall at object
  - say “Put that”… point to destination… “there”
- Example: eye-driven window interface

7. Bandwidth

- Old: Low input bandwidth (keys, mouse)
  - New: Very high bandwidth
- Systems may incorporate motion tracking, virtual reality, speech, “peripheral” input
- Difficulties
  - requires lots of processing power, both for accepting input and interpreting it
  - lags are unacceptable!
    (e.g., motion sickness in virtual environments)

8. Tracking feedback

- Old: Feedback only after completed input
- New: Continuous feedback on-the-fly
- Example: Emacs search (sort of)
- Example: movie database
- Interface should react like the real world
  - again, input-output lag is an issue
  - again, processing power is an issue
9. Turn-taking
- Old: First user, then computer, then user...
  New: Continuous stream for both
- Closely related to feedback... computer is always responding, so there’s never a “turn”
- How does this map onto the real world?
  - when we interact with the inanimate world?
    - e.g., walking through the park, playing basketball
  - when we interact with the animate world?
    - e.g., talking to someone, interacting with a pet
  - we sometimes take turns in the real world... why not in a user interface?

10. Interface locus
- Old: Computer on the desk
  New: Computers everywhere
- Ubiquitous computing from...
  - smaller, lighter “computers”
    - e.g., PDAs, calculators, watches
  - computers built into everyday objects
    - e.g., ovens, cars, shopping carts
  - computers built into not-so-everyday objects
    - e.g., pet dog robots
- Good for awareness, “telepresence”; dangerous for privacy?

11. User programming
- That is, programming for end users
- Old: (Usually) hard-core macro languages
  New: Smooth adaptation of objects
- Example: object-oriented customization
  - “take your basic <object> “but make the” <subobject> “behave like this...”
  - if possible, can specify with state transitions
    - like storyboards, or like SILK’s behaviors
- BUT in the end, complex programs require complex languages
  - hard (for me) to envision huge successes here

12. Software packaging
- Old: Application-centered approach
  New: System-wide, OO approach
- Example: spell checkers
  - should be only one for your entire system
  - does Microsoft have this right??
    - integrated across Word, Excel, PowerPoint, email
    - easier to do for a single vendor, and has the unfortunate side effect of monopolization
    - open source, open standards — but can we really arrive at a true standard?
Sample noncommand interface domains

- Eye-tracking interfaces
  - control by gaze, explicitly or implicitly
- Computer music
  - computer listens, plays along, harmonizes, etc.
- Interface agents
  - provide active help, reminders, etc.
- Embedded help
  - actually show the process, guiding the user’s “hand” to the right places

Gaze-Added Interfaces (GAIs)

- Users control the interface using gaze and/or other inputs
- Gaze added to basic/existing input instead of replacing it
- Users can...
  - employ only basic inputs
  - employ only gaze input
  - employ any combination of basic / gaze

HC “Eye”

- Human --> machine communication...
  - keyboard, mouse
  - speech, handwriting, gestures
  - poking (CRL Kiosk), tickling (Tickle-Me Elmo) !!
- Gaze-based interfaces (GBIs)
  - users control the interface using gaze / eye movements
  - typical focus on disabled users
  - gaze is often the only (primary) input

Case Study: IGO
Gaze-Added Input

- Gaze focus: background highlight
- Gaze button: (keyboard) key
- Gaze control analogous to mouse
  - click for select
  - double-click for open
  - hold for drag

Intelligent Interpretation

- All GBIs must interpret gaze - i.e., assign gaze to intended object
- Standard interpretation
  - assign gaze to underneath / nearest object
- Intelligent interpretation
  - assign gaze to most likely object
  - use probabilistic model of behavior

Intelligent Interpretation

- Find object that maximizes
  \[ \Pr(gaze|object) \cdot \Pr(object|history) \]

Gaussian distribution of gaze location \((x,y)\) around object center

Distribution of object probabilities given some action history

File   Edit   Help

.20 .40 .10

Intelligent Interpretation

Normal   After Icon Select

Copy Paste

Edit

Copy Paste
**Evaluation Study**

- **Training Stage** (8 blocks of 10 trials)
  - gaze only or mouse only
- **Free Stage** (2 blocks of 10 trials)
  - gaze and/or mouse as desired
- **Trial** = one of five tasks (5-10 s)
- **Ten users with no GBI experience**
- **Equipment**
  - IScan head-mounted eye tracker

**Task Times**

- **No difference between gaze and mouse**

**Task Errors**

- **More errors with gaze than mouse**
- **Errors in free stage -> gaze use**

**Free Stage Gaze Use**

- **Overall gaze use = 67%**
- **Correlated to (gaze-mouse) times (R=.70)**
**Gaze Use by Action Type**

- Less complex actions -> more gaze use

![Gaze Use by Action Type Chart]

**Intelligent Interpretation**

- Comparison of correctness with...
  - intelligent interpretation
  - “no-context” interpretation: assign gaze to nearest object
  - “basic” interpretation: assign gaze to underneath object

<table>
<thead>
<tr>
<th>Intelligent</th>
<th>No-Context</th>
<th>Basic</th>
</tr>
</thead>
<tbody>
<tr>
<td>83%</td>
<td>65%</td>
<td>17%</td>
</tr>
</tbody>
</table>

**Lessons Learned**

- Gaze nicely complements other inputs
  - users quickly adapt to gaze input
  - users successfully interleave gaze, mouse
- Common difficulties
  - “leave before click”
  - gaze dragging
  - handling two “cursors”
- Intelligent interpretation helps
  - eye trackers will improve, but variability will remain
  - better eye tracking -> greater usability

**Thought question...**

- Ok, let’s try to put all this to use.
- We know how a typical web browser looks
- Can we design a new one “beyond WIMP”?