CS 338: Graphical User Interfaces

Lecture 9-1: 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!

Evolution of computers (& users)

Generation	Hardware technology	Operating mode	Programming languages	Terminal technology	User types	Advertising image	User interface paradigm
1 1945-1955 Pioneer	Vacuum tubes, huge machines, much cooling, short mean time between failures.	One user at a time "owns" machine (but for a limited time only)	Machine language 001100111101	TTY, typewriter. Only used in the computer center.	Experts, pioneers	Computer as calculator	Programming
2 1955-1965 Historical	Transistors; more reliable. Computers start seeing use outside the lab.	Batch ("computer as temple" -make offerings to get oracle replies)	Assembler ADD A,B	Line-oriented terminals ("glass-TTY")	Technocrats, professional computerists	Computer as information processor	Command languages
3 1965-1980 Traditional	Integrated circuits. Businesses can cost-justify buying computers for many needs.	Timesharing (online transaction processing systems)	"High-level" languages, Fortran, Pascal	Full screen terminals, alphanumeric characters only. Remote access common.	Specialized groups without computer knowledge (e.g. bank tellers)	Mechanization of white-collar labor	Full-screen strictly hierarchical menus and form fill-in
4 1980-1995 Modern	VLSI. Individuals can buy their own personal computer	Single user personal computers	Problem oriented languages, spreadsheets	Graphical displays with fair resolution. Desktops and heavy portables.	Business professionals, hobbyists	Personal productivity (computer as tool)	WIMP (Windows, Icons, Menus, and a Pointing device)
5 1996-? Future	Wafer-scale integration, computer-on-a-chip. Individuals can buy *many* computers.	Networked single user systems and embedded systems	Non-imperative, possibly graphical	"Dynabook" [61], multimedia I/O, easily portable, with cellular modem.	Everybody	Computer as entertainment	Noncommand interfaces

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

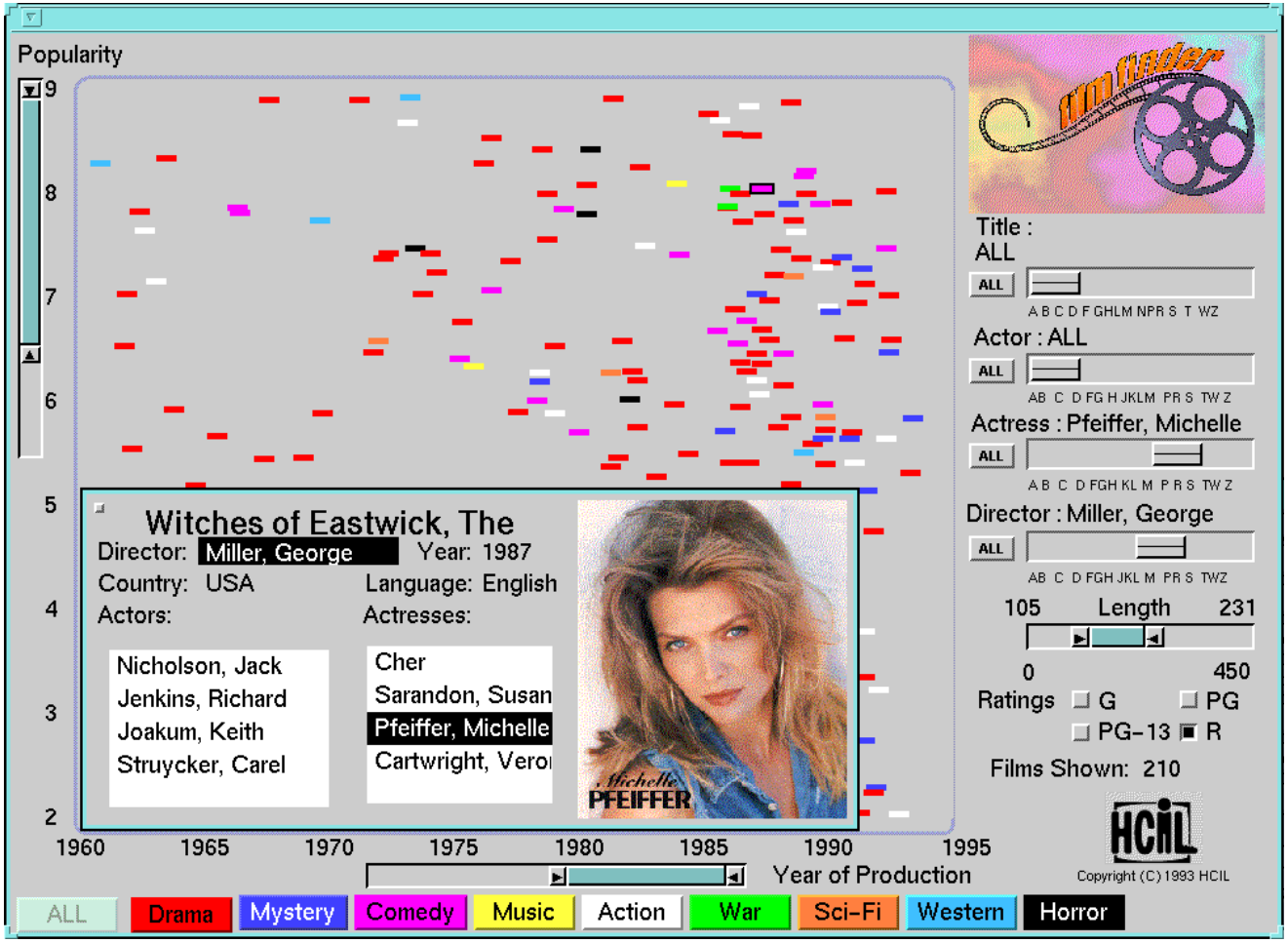
- "Old-school" interfaces are functional in requiring specification of entire function
  - uses a "verb-noun" syntax
  - e.g., "rm foo", "emacs file.java"
- Current GUIs are object-oriented
  - uses a "noun-verb" syntax
  - e.g., select icon, drag to trash / select "Open"
- What seems to be coming on 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

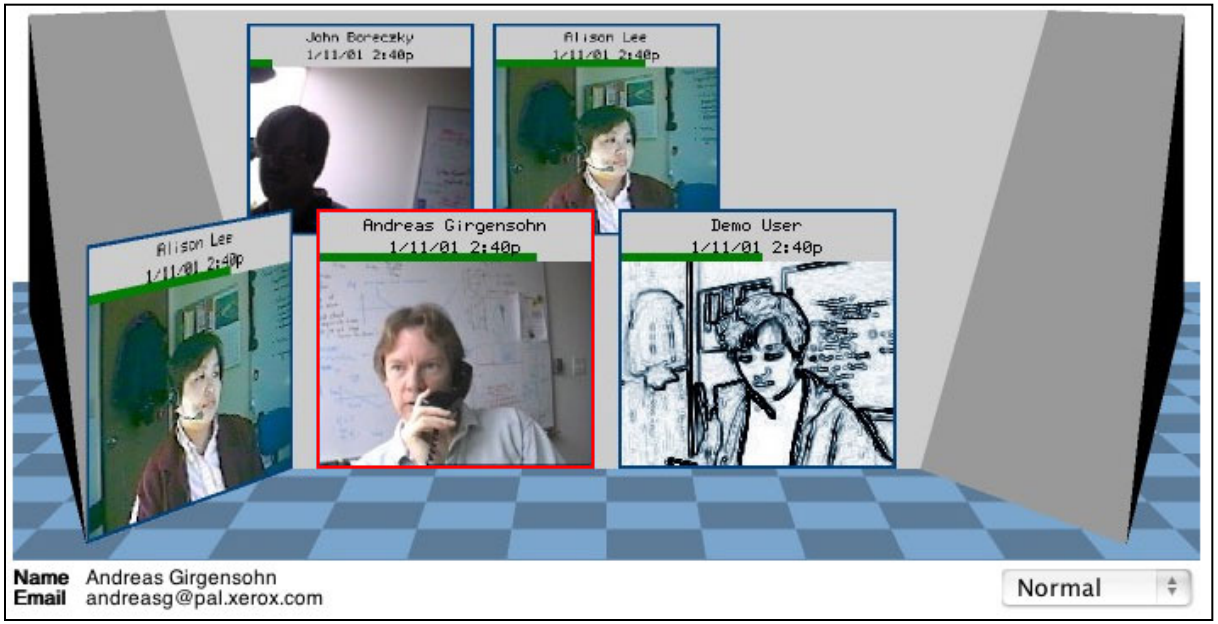


Non-command interfaces

- 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

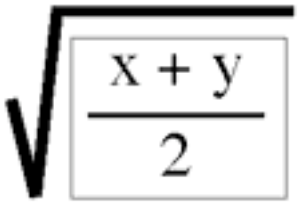
- Old: "Do what you're told"
  New: "Do what I mean!"
- Great idea... but why is this hard?
- Example: Intelligent tutoring
  - monitor what the student knows
  - interrupt with instruction when necessary
- Example: Model tracing / "Mind tracking"
  - infer student knowledge, or disabled user's intentions,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

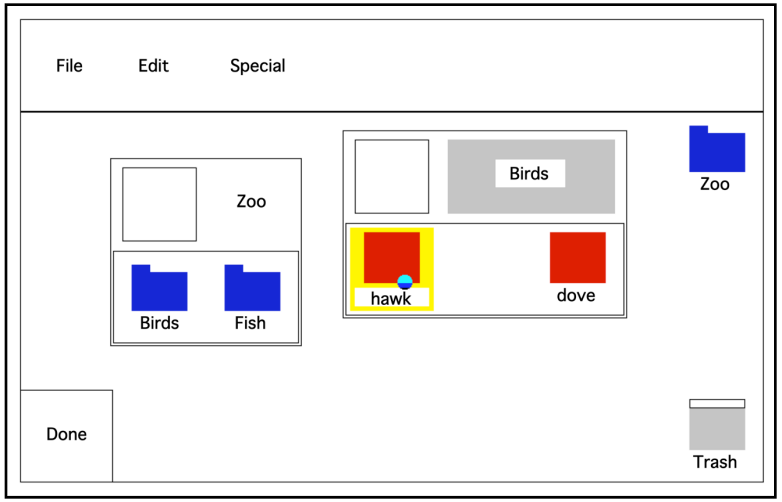


5. Object visibility

- 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
  - 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?

Some noncommand domains

- Interface agents
  - provide active help, reminders, etc.
- Embedded help
  - actually show the process, guiding the user's "hand" to the right places
- Computer music
  - computer listens, plays along, harmonizes, etc.

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"?